PC-HELPER

絶縁型デジタル入出力ボード

PIO-32/32L(PC)

解説書

CONTEC

確認してください！

本商品は次のような構成になっています。

構成品がすべて揃っていることを確認してください。万一、構成品が足りない場合や破損している場合は、お買い求めの販売店にご連絡ください。

登録カードは、新商品情報などを皆さまにお知らせする際に必要なカードです。ご記入の上、必ずご返送くださいますようお願いします。

◆商品構成

- □PIO-32/32L(PC)ボード…1
- □解説書…1
- □登録カード&保証書…1
- □登録カード返送用封筒…1
- □Question用紙…1

サポート体制

当社商品をよりよく、より快適にご使用いただくために、次のようなサポートを行っております。

・インフォメーションセンターの設置
・各種セミナーの開催
・FA/LA無料相談コーナー
・システム受託開発、OEM受託

「インフォメーションセンター」は当社商品に関する技術的な問い合わせに対し、電話とFAXで対応する窓口です。商品のシリーズごとに専門のスタッフが対応しますので、Question用紙(商品添付)の裏面をご覧の上、該当するインフォメーションセンターまでFAXでお問い合わせください。

「FA/LA無料相談コーナー」は、皆さまがシステムを構築する際に当社商品の選定の相談をお受けする窓口です。面談によるシステム相談を専門スタッフが担当いたします。お問い合わせは、当社営業所までご連絡ください。

また商品のカタログや概要書の請求は、お買い求めの販売店または当社営業所までご連絡ください。

ご注意

①本書の内容の全部、または一部を無断で転載することは禁止されています。

②本書の内容に関しては将来予告なしに変更することがあります。

③本書の内容については万全を期しておりますが、万一ご不審な点や記載もれなどお気づきのことがありましたら、お買い求めの販売店へご連絡ください。

④本商品の運用を理由とする損失、逸失利益などの請求につきましては、前項にかかわらず、いかなる責任も負いかねますのであらかじめご了承ください。

⑤MS、Microsoft、MS-DOSは、米国Microsoft Corporationの登録商標です。
Windowsは、米国Microsoft Corporationの登録商標です。
その他、本書中に使用している会社名および商品名は、一般に各社の商標または登録商標です。

取り扱い上の注意

このボードを取り扱うときは、次のことに十分ご注意ください。

・ボードに衝撃を与えたり、曲げたりしないでください。破損の原因となります。

・ボードを改造しないでください。改造をしたものに対しては、当社は一切の責任を負いません。

・ボードの金メッキ端子部(エッジコネクタ)には手を触れないでください。接触不良の原因となります。触れた場合は、工業用アルコールできれいにふいてください。

・ボード上には、あらかじめ設定を必要とするスイッチやジャンパがあります。スロットに実装する前に必ず確認してください。

・ボード上のスイッチやジャンパは指定以外の設定にしないでください。故障の原因となります。

・ボードはパソコン本体の拡張スロット、あるいは当社製I/O拡張ユニットに実装して使用してください。

・パソコン本体の電源が入った状態で、ボードをスロットに挿入したり、抜いたりしないでください。故障の原因になります。必ずパソコン本体の電源を切ってから行ってください。また、I/O拡張ユニットで使用するときも同様に、ユニットの電源を切ってから行ってください。

・パソコンの拡張スロットに実装するボードの消費電流の総和は、パソコンの許容電源容量より大きくなってはいけません。故障の原因となります。許容電源容量より大きくなる場合は、I/O拡張ユニットを増設するなどして許容電源容量内で使用してください。

はじめに

このたびは、PIO-32/32L(PC)をご購入いただきまして、ありがとうございます。
この製品は、絶縁型デジタル入出力を行うPC/ATおよびその互換機用のインターフェイスボードです。このボードは、パソコン本体の拡張スロットまたはI/O拡張ユニットに実装して使用し、これ1枚で最大32点入出力することができます。このボードは入力に電流ソースタイプ、出力に電流シンクタイプとなっています。

◆特長

・フォトカプラによる絶縁入出力で耐ノイズ性が向上しています。

・8点を1グループとして4グループ、32点のデジタル電流ソースタイプ信号が入力できます。

・8点を1グループとして4グループ、32点のデジタル電流シンクタイプ信号が出力できます。

・入力信号32点のうち4点を、割り込み入力として最大4レベルの同時割り当てができます。

・出力定格は、1点当たり最大DC35V、150mA(ただし、1コモン当たり最大2A)の大容量に設計されています。

・汎用の入出力機能のほかに、オプション機能を用意しています。

例) ・入力信号にデジタルフィルタをかけることができます。
・出力データをモニタすることができます。
・STB/ACK信号によるデータの通信を行うことができます。
・ビット単位でデジタル入出力を行うことができます。

◆機能

■汎用入出力機能

・入力

このボードは、外部から与えられた最大32点のデジタル信号を、8点単位で構成されるグループで読み出し、ボードを装着したパソコンに送出します。パソコンからのこのボードに対するアクセスは、任意に設定できる4つの入力ポートを介して行います。IN命令の実行によってこれらの入力ポートを読み出すと、その入力ポートに該当するバッファゲートが開かれ、外部装置から与えられているデジタル信号がグループ単位で取り込まれます。このとき、パソコンに送出される信号は負論理となります。また、入力32点のうち4点は、パソコンのユーザー解放割り込み入力として割り当てることができますので、この信号を割り込み要求信号として使用できます。

・出力

このボードは、最大32点のデジタル信号を、8点単位で構成されるグループで外部装置に書き出します。パソコンからのこのボードに対するアクセスは、任意に設定できる4つの出力ポートを介して行います。OUT命令の実行によって出力ポートにデータを書き出すと、その出力ポートに該当するラッチ回路にデータが保持されます。そして、デジタル信号はフォトカプラにより電気的に絶縁された後、トランジスタを通じて、接続されている外部装置にグループ単位で送出されます。このとき、外部装置に送出される信号は負論理となります。また、ラッチ回路上のデータは、再度OUT命令が実行されるまでその状態が保持されます。

■オプション機能

・ビット単位の入出力機能

I/Oアドレスを意識しないで、任意の1点(1ビット)に対してデータを入出力できます。

・グループ単位の入出力機能

I/Oアドレスを意識しないで、任意の1グループ(8点単位)に対してデータを入出力できます。

・デジタルフィルタ機能

入力信号のノイズやチャタリングを、ハードウェア的に防止できます。

・出力データのモニタ機能

現在出力している信号データがわかります。

・ハンドシェイク機能

STB/ACK信号を用いたデータの通信が行えます。

・割り込みコントロール機能

オプション機能使用時にハードウェア割り込みが利用できます。

◆本書の構成

この解説書は次のような構成になっています。

第1章　ボードの設定
ボードのスイッチの設定方法を説明しています。

第2章　外部機器との接続
インターフェイスコネクタや外部入出力回路

第3章　I/Oポートのビット割り付け
ボードが使用しているI/Oポートのビット割り付けおよび各ビットの定義について説明しています。

第4章　オプション機能
オプション機能の概要とその使用方法を説明しています。

第5章　ボード本体の仕様
ボード本体の仕様や回路ブロック図などを説明しています。

付録　ボードを使用する上での参考資料をまとめています。必要に応じて参照してください。

目次

はじめに
◆特長
◆機能
◆本書の構成

第1章　ボードの設定

ボード本体各部の名称 1-2
I/Oアドレスの設定 1-3
◆注意事項 1-3
◆設定方法 1-4
割り込みレベルの設定 1-5
◆注意事項 1-5
◆設定方法 1-5
使用する機能の設定 1-8
◆注意事項 1-8
◆設定方法 1-8

第2章　外部機器との接続

インターフェイスコネクタの説明 2-2
◆インターフェイスコネクタの接続方法 2-2
◆インターフェイスコネクタの信号配置 2-3
◆PCB96WSおよびCCB-96信号配置 2-4
外部入出力回路 2-5

第3章　I/Oポートのビット割り付け

I/Oポートのビット割り付け<汎用入出力> 3-2
◆入力ポートのビット割り付け 3-2
◆出力ポートのビット割り付け 3-3

第4章　オプション機能

こんなときにはオプション機能を！ 4-2
オプション機能を使うとき 4-3
◆オプション機能を使うためには 4-3
◆I/Oポートのビット割り付け＜オプション機能＞ 4-4
◆インターフェイスコネクタの信号配置＜オプション機能＞ 4-6
オプション機能の使用方法 4-7
◆ビット単位の入出力機能 4-7

◆グループ単位の入出力機能 4-9
◆デジタルフィルタ機能 4-11
◆出力データのモニタ機能 4-13
◆ハンドシェイク機能 4-14
◆割り込みコントロール機能 4-16

第5章　ボード本体の仕様

回路ブロック図 5-2
仕様 5-3

付録

A. PC/ATシリーズでの割り込み 2
B. LSIに対するリカバリタイム 15
C. サンプルプログラム 16
D. サージ電圧の対策 33
E. 実装方法 34

第1章

ボードの設定

●ボード本体各部の名称…1-2
●I/Oアドレスの設定…1-3
●割り込みレベルの設定…1-5
●使用する機能の設定…1-8

ボード本体各部の名称

ボード本体の各部の名称を図1.1に示します。

なお、図中のスイッチとジャンパの状態は、出荷時の設定を示しています。

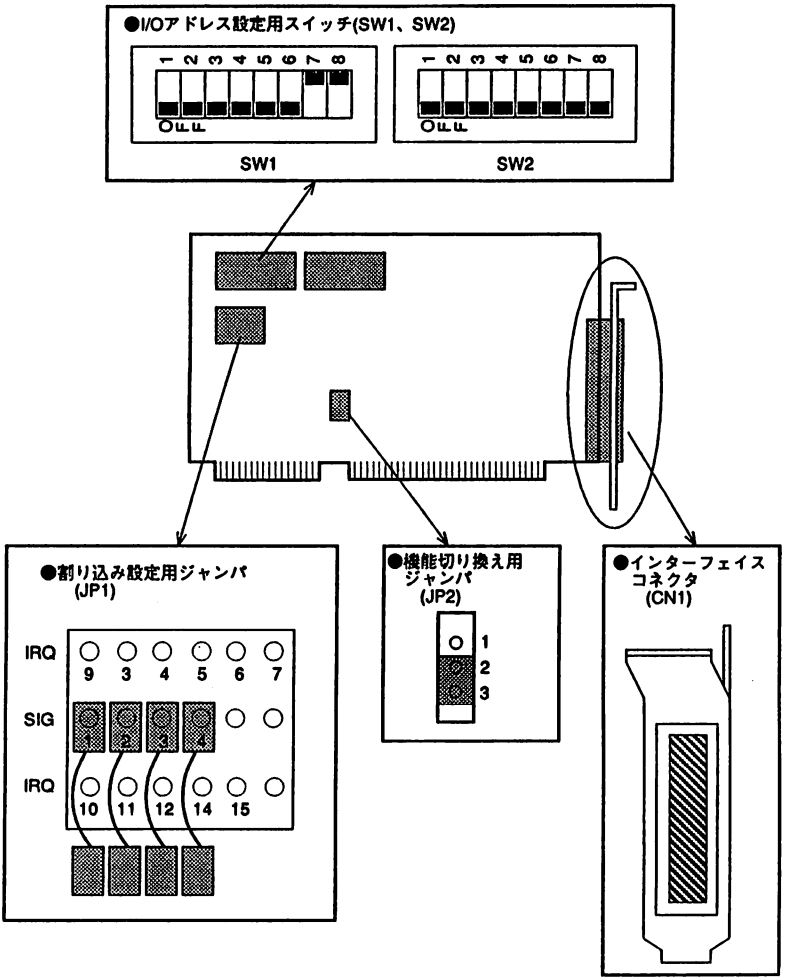


図1.1　各部の名称と出荷時設定

I/Oアドレスの設定

このボードはパソコンの入出力命令によってコントロールされる装置(I/Oデバイス)です。I/Oデバイスにはパソコン内部の装置や拡張ボードなどがあり、これらを区別する番号がI/Oアドレスです。I/Oアドレスは0300Hのように4桁の16進数で表現し、各I/Oデバイスの認識番号を示します。

多くの拡張ボードは、連続する複数のI/Oアドレスを使ってコントロールします。連続するI/Oアドレスの最初の値を先頭I/Oアドレスといいます。
このボードでは、汎用入出力の場合連続する4ポート分のI/Oアドレスを使用します。

◆注意事項

PC/ATとその互換機では、[0000H～FFFFH]の範囲のI/Oアドレスに対して入出力命令を実行することによって、ハードウェアを動作させることができます。ところが、これらのパソコンではテクニカルマニュアルなどのI/Oアドレスマップの項を見てもわかるとおり、パソコン本体内でCRTやキーボード、その他のコントロール用としてあらかじめ特定のI/Oアドレスがすでにシステムで使用済みか予約済み(Reserved)となっており、ユーザーの使用が禁止されています。
ユーザーが使用できるI/Oアドレスは制限されていますが、コンテックでは図1.2に示す範囲を推奨しています。

推奨I/Oアドレス

＊300H～＊31FH	
＊700H～＊71FH	
＊B00H～＊B1FH	
＊F00H～＊F1FH	(＊：0～Fの任意の値)

図1.2　推奨I/Oアドレス

これはI/Oアドレスの下位3桁(16進数)についての制限ですが、上位1桁については0～Fの範囲から任意に値を選択することができます。

複数の拡張ボードを使用する場合は、それぞれのボードが占有するI/Oアドレスが重複しないように設定してください。

LANボードが実装済みのパソコンの場合300H～31FHまですでに使用されていますので、I/Oアドレスが重ならないようにしてください。

◆設定方法

先頭I/Oアドレスは、ボード上のディップスイッチ(SW1, SW2)で設定します。SW1とSW2の各ビットは先頭I/Oアドレスの上位14ビット(A15～A2)に対応しており、A1～A0は常に「0(OFF)」に固定してください。

SW1とSW2の各ビットのON、OFFは先頭I/Oアドレスを２進数に変換した値を示し、ONが「1」に対応し、OFFが「0」に対応します。

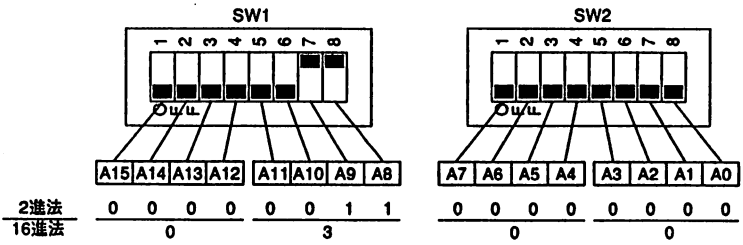


図1.3　I/Oアドレスの設定

図1.3では先頭I/Oアドレスが0300Hに設定されており、表1.1に示すI/Oアドレスがこのボードによって占有されます。

表1.1　占有するI/Oアドレス例

使用する機能	汎用入出力機能	オプション機能
占有するI/Oアドレス	0300H～0303H (4ポート)	0300H～0307H および 1300H～1307H (16ポート)

割り込みレベルの設定

このボードでは、入力信号32点のうち4点のデジタル信号およびオプション機能使用時のハンドシェイク信号などを、割り込み要求信号として使用することができます。この信号によって、割り込み要求がパソコンへ出力されますので、パソコンの割り込み機能を利用することができます。割り込み設定はボード上のジャンパ(JP1)で行います。

割り込みを使用しないときは、入力信号を特定レベルに接続しないように、リード線付きショートコネクタで設定してください。

割り込みを使用するときは、ボード上のジャンパ(JP1)で割り込みレベルを設定します。このボードで設定できる割り込みレベルはIRQ3～7、9～12、14、15ですが、パソコン本体およびほかのボードで使用されていないレベルに設定してください。割り込み要求信号は、入力信号と一対一の対応で最大4つのレベルまで割り当てることができます。

◆注意事項

①割り込みを使用する場合、他で使用している割り込みレベルと重複しないように設定してください。

②電源が入っているパソコン(またはI/O拡張ユニット)にこのボードが実装されているときは、JP1のショートコネクタは抜き差ししないでください。

◆設定方法

割り込みレベルは、ボード上のジャンパ(JP1)で設定します。

■割り込みを使用しないとき

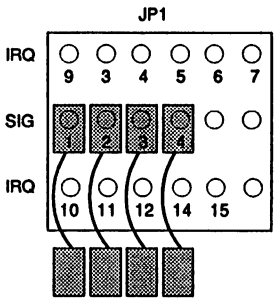


図1.4　割り込みジャンパのアクセサリーパーツ

■割り込みを使用するとき

割り込みレベルをショートコネクタで接続してください。設定できる割り込みレベルはIRQ3～7、9～12、14、15です。ただし、XTバス仕様のパソコンではIRQ10～15は使用できません。

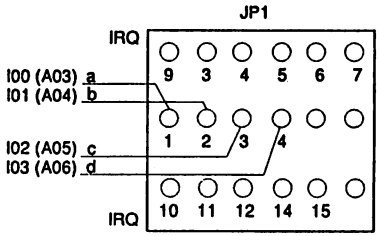


図1.5　汎用入出力機能使用時の割り込み設定

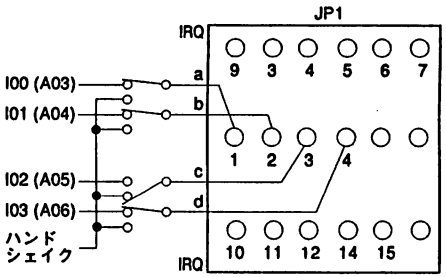
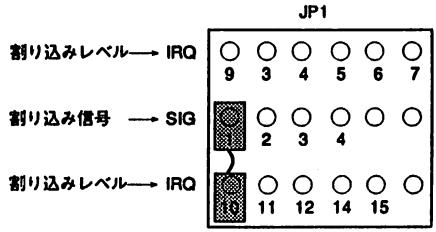


図1.6　オプション機能使用時の割り込み設定

(例) インターフェイスボードで「SIG1(I00)」に割り込みをいかし、PC/ATのIRQ10に割り込みをつなぐ場合以下のようになります。(汎用入出力)



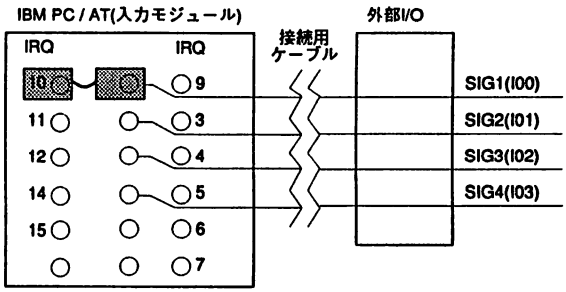


図1.7 汎用入出力時の割り込み設定例

使用する機能の設定

このボードには、デジタル信号の汎用入出力機能のほかに、オプション機能が用意されています。

①汎用入出力機能
デジタル信号(最大32点)を入力および出力します。また、入力信号のうち4点を割り込み信号として利用することができます。
②オプション機能
デジタル信号の入出力以外に、次のようなことを行うことができます。
・入力信号にデジタルフィルタをかける。
・出力データをモニタする。
・STB/ACK信号を利用して、データを入出力する。
など

ボード上のジャンパ(JP2)を用いて、使用する機能を設定してください。

◆注意事項

①JP2は指定以外の設定にしないでください。

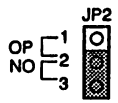
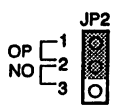
②電源が入っているパソコン(またはI/O拡張ユニット)にこのボードが実装されているときは、JP2のショートコネクタは抜き差ししないでください。

③JP2の設定によって、ボードの機能はもちろん、占有するI/Oアドレスや I/Oポートのビット割り付け、コネクタの信号配置も変わります。設定を変更するときは、これらの点にも注意してください。

◆設定方法

使用する機能は、ボード上のジャンパ(JP2)で設定します。

表1.2　使用する機能の設定

機能	設定
汎用入出力機能 (NORMAL)	
オプション機能 (OPTION)	

第2章

外部機器との接続

●インターフェイスコネクタの説明…2-2
●外部入出力回路…2-5

インターフェイスコネクタの説明

◆インターフェイスコネクタの接続方法

このボードと外部機器との接続は、ボード上のインターフェイスコネクタ(CN1)で行います。

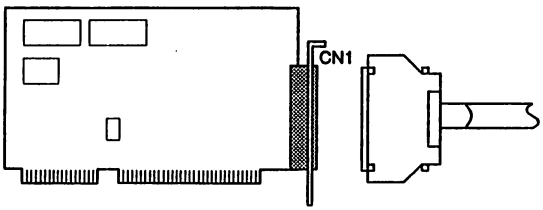
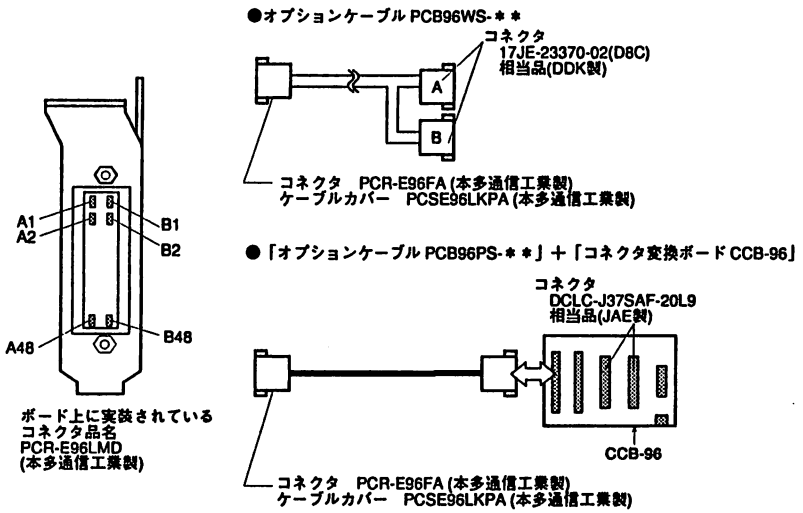


図2.1　インターフェイスコネクタの接続

・使用コネクタ
PCR-E96LMD [本多通信工業製]

・適合コネクタ
PCR-E96FA [本多通信工業製]

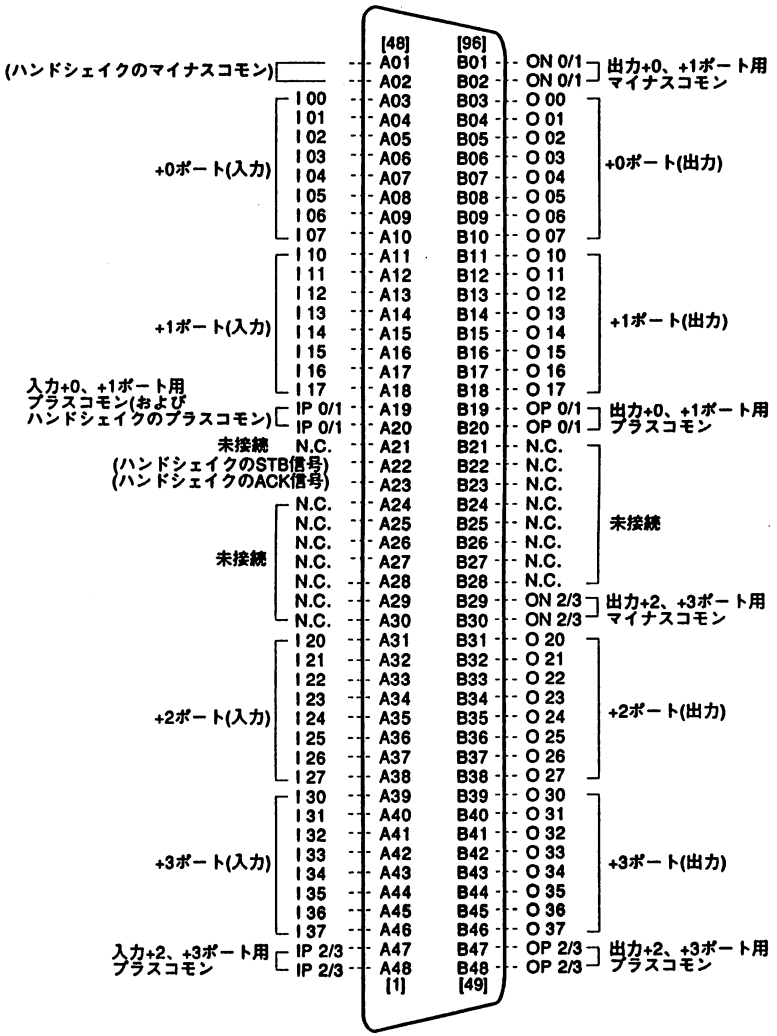


＊＊はケーブル長です(1.5m、3m、または5m)

図2.2　使用コネクタ

◆インターフェイスコネクタの信号配置

本インターフェイスボードと外部装置の接続は、ボードに実装された96ピンのコネクタで行います。



・[ ]内は本多通信工業(株)指定の端子番号です。

図2.3 インターフェイスコネクタの信号配置

◆PCB96WSおよびCCB-96信号配置

オプションケーブルと各信号の対応は以下の通りです。

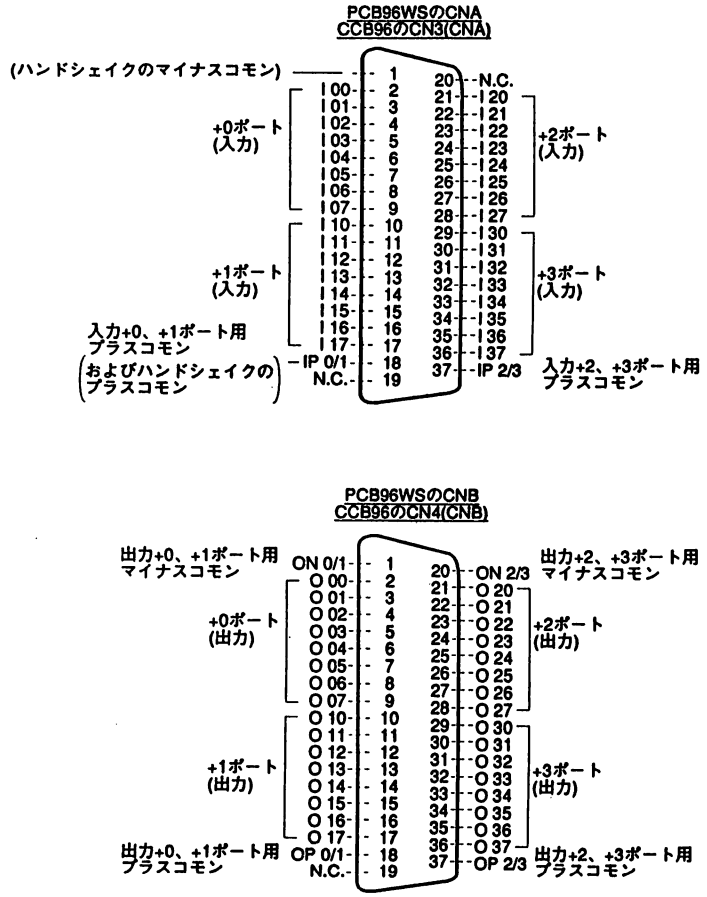


図2.4　PCB96WSおよびCCB-96の信号配置

外部入出力回路

■入力部

このボードのインターフェイス部の入力回路は、図2.5の通りです。信号入力部は、フォトカプラ絶縁による電流ソースタイプ入力になっています。したがって、このボードの入力部を駆動するためには外部電源が必要です。このとき必要となる電源容量は、DC24V時入力1点当たり約8mA(DC12V時には約4mA)です。なお、DC24V使用時に「ON」となる入力点数が多い場合、入力抵抗で若干の発熱を生じます。

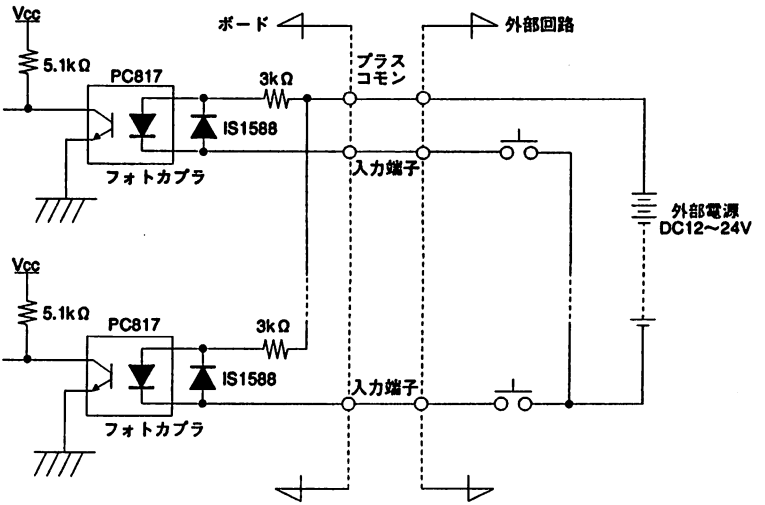


図2.5　入力回路

■出力部

このボードのインターフェイス部の出力回路は、図2.6の通りです。信号出力部はフォトカプラ絶縁による電流シンクタイプ方式になっています。したがって、このボードの出力を駆動するためには外部電源が必要です。出力電流の定格は1点当たり最大150mA、1コモン当たり最大2Aです。また、このボードの出力トランジスタには、サージ電圧保護回路が付加されていません。したがって、このボードでリレーやランプなどの誘導負荷を駆動する場合には、負荷側でサージ電圧対策を行ってください。サージ電圧対策については、「付録D. サージ電圧の対策」を参照してください。

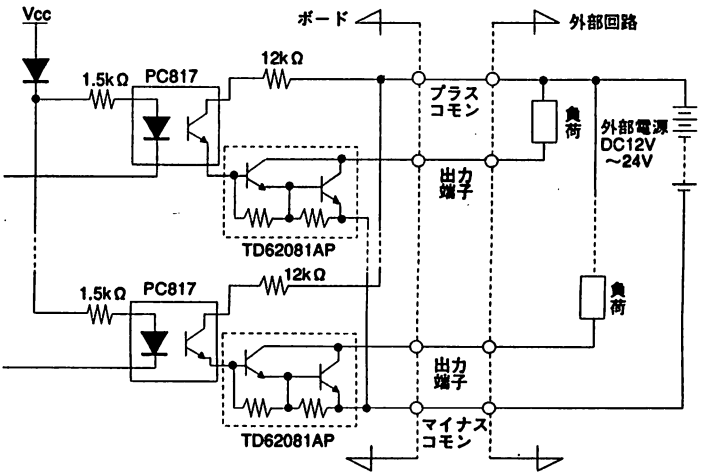


図2.6　出力回路

■STB、ACK信号の入出力部

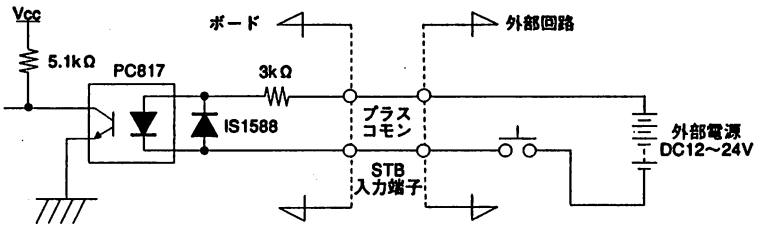


図2.7　STB入力回路

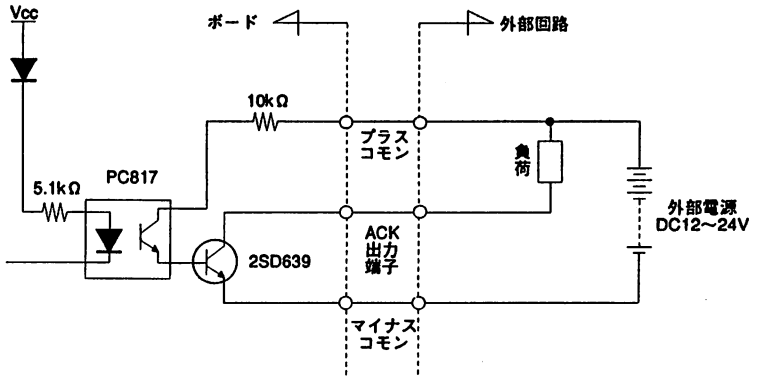


図2.8　ACK出力回路

第3章

I/Oポートのビット割り付け

●I/Oポートのビット割り付け
<汎用入出力>…3-2

I/Oポートのビット割り付け<汎用入出力>

汎用入出力機能使用時に、パソコンからのこのボードに対するアクセスは、4つのI/Oポートを介して行われます。ボード上のコネクタに接続される外部のデジタル信号(入力信号および出力信号)が、8点を1グループとして各ポートに割り当てられています。

◆入力ポートのビット割り付け

入力ポートのビット割り付けは、次のようになっています。

先頭I/Oアドレス		D7	D6	D5	D4	D3	D2	D1	D0
+0H		入力グループ0 (+0ポート)							
		I07 [A10]	I06 [A09]	I05 [A08]	I04 [A07]	I03 [A06]	I02 [A05]	I01 [A04]	I00 [A03]
+1H		入力グループ1 (+1ポート)							
		I17 [A18]	I16 [A17]	I15 [A16]	I14 [A15]	I13 [A14]	I12 [A13]	I11 [A12]	I10 [A11]
+2H		入力グループ2 (+2ポート)							
		I27 [A38]	I26 [A37]	I25 [A36]	I24 [A35]	I23 [A34]	I22 [A33]	I21 [A32]	I20 [A31]
+3H		入力グループ3 (+3ポート)							
		I37 [A46]	I36 [A45]	I35 [A44]	I34 [A43]	I33 [A42]	I32 [A41]	I31 [A40]	I30 [A39]

表中のI××は信号名を示し、[ ]内はコネクタの端子番号を示します。
また、I00、I01、I02、I03は割り込み信号としても使用できます。

図3.1 入力ポートのビット割り付け

入力データが「ON」のとき、該当するビットは「1」になります。逆に入力データが「OFF」のときは、「0」になります。

例)先頭I/Oアドレスが300Hのとき、I07が「ON」かどうか調べる場合。

①BASIC (MS-DOS版)

```
DATA%=INPUT(&H300)
IF(DATA% AND &H80)=&H80 THEN …
                :
                :
```

②Microsoft C / C++ (MS-DOS版)

```
data_in=inp(0x300);
if(data_in & 0x80) …
                :
                :
```

◆出力ポートのビット割り付け

出力ポートのビット割り付けは、次のようになっています。

先頭I/Oアドレス		D7	D6	D5	D4	D3	D2	D1	D0
	+0H	出力グループ0 (+0ポート)							
		O 07 [B10]	O 06 [B09]	O 05 [B08]	O 04 [B07]	O 03 [B06]	O 02 [B05]	O 01 [B04]	O 00 [B03]
	+1H	出力グループ1 (+1ポート)							
		O 17 [B18]	O 16 [B17]	O 15 [B16]	O 14 [B15]	O 13 [B14]	O 12 [B13]	O 11 [B12]	O 10 [B11]
	+2H	出力グループ2 (+2ポート)							
		O 27 [B38]	O 26 [B37]	O 25 [B36]	O 24 [B35]	O 23 [B34]	O 22 [B33]	O 21 [B32]	O 20 [B31]
	+3H	出力グループ3 (+3ポート)							
		O 37 [B46]	O 36 [B45]	O 35 [B44]	O 34 [B43]	O 33 [B42]	O 32 [B41]	O 31 [B40]	O 30 [B39]

表中のO ××は信号名を示し、[ ]内はコネクタの端子番号を示します。

図3.2 出力ポートのビット割り付け

該当するビットに「1」を出力すると、出力データが「ON」になります。逆に「0」を出力すると、「OFF」になります。

例)先頭I/Oアドレスが300Hのとき、O07のみを「ON」にする場合。

①BASIC (MS-DOS版)

```
OUT &H300,&H80
```

②Microsoft C / C++ (MS-DOS版)

```
outp(0x300,0x80);
```

第4章

オプション機能

●こんなときには
オプション機能を！…4-2
●オプション機能を使うとき…4-3
●オプション機能の使用方法…4-7

こんなときにはオプション機能を！

①I/Oアドレスを意識しないで、任意の1点(1ビット)に対してデータを入出力したい。
⇨ビット単位の入出力機能

②I/Oアドレスを意識しないで、任意の1グループ(8点単位)に対してデータを入出力したい。
⇨グループ単位の入出力機能

③入力信号のノイズやチャタリングをハードウェア的に防止したい。
⇨デジタルフィルタ機能

④現在出力している信号データを知りたい。
⇨出力データのモニタ機能

⑤STB/ACK信号を用いて、データの通信を行いたい。
⇨ハンドシェイク機能

⑥オプション機能使用時にハードウェア割り込みを利用したい。
⇨割り込みコントロール機能

オプション機能を使うとき

◆オプション機能を使うためには

オプション機能を使うためには、JP2の1-2間をショート状態にすることが必要です。

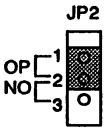


図4.1　オプション機能使用時のJP2設定状態

なお、商品出荷時は2-3間がショートされています。JP2の詳細は、「第1章　使用する機能の設定」を参照してください。

▼注意

電源が入っているパソコン(または拡張ユニット)にこのボードが実装されているときは、JP2のショートコネクタは抜き差ししないでください。

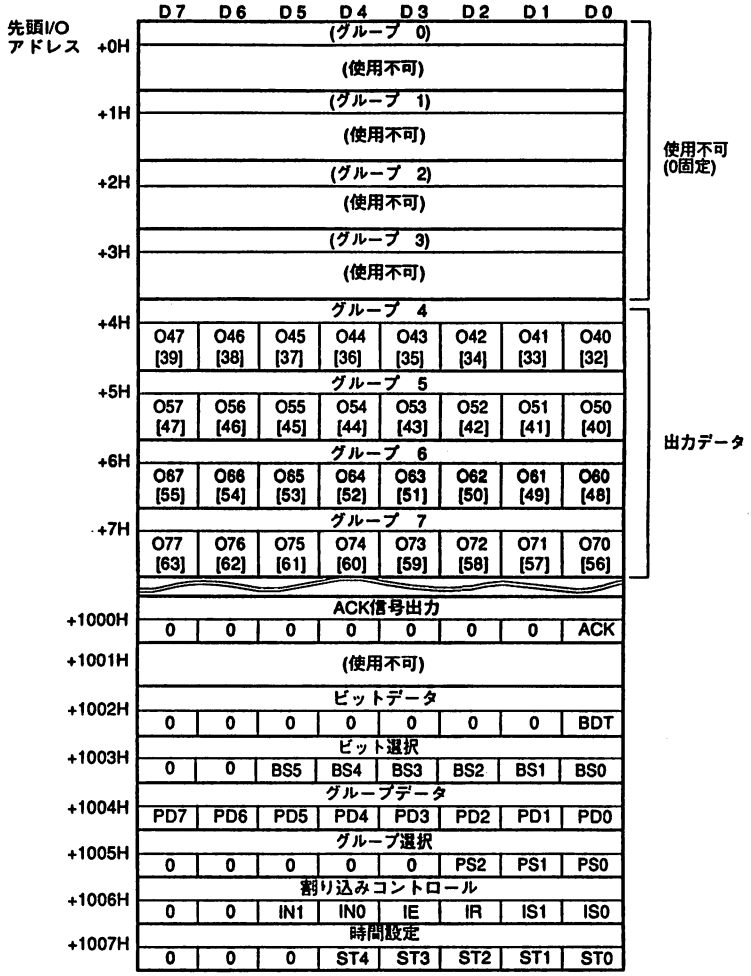
◆I/Oポートのビット割り付け<オプション機能>

オプション機能を使用する場合のI/Oポートのビット割り付けを図4.2、図4.3に示します。なお、「使用不可」のポートは、ボードのコントロールには関係ありませんが、このボードによって占有されているため使用することはできません。

先頭I/Oアドレス		D7	D6	D5	D4	D3	D2	D1	D0	
	+0H	グループ 0								入力信号用
		I07 [7]	I06 [6]	I05 [5]	I04 [4]	I03 [3]	I02 [2]	I01 [1]	I00 [0]	
	+1H	グループ 1								
		I17 [15]	I16 [14]	I15 [13]	I14 [12]	I13 [11]	I12 [10]	I11 [9]	I10 [8]	
	+2H	グループ 2								
		I27 [23]	I26 [22]	I25 [21]	I24 [20]	I23 [19]	I22 [18]	I21 [17]	I20 [16]	
	+3H	グループ 3								
		I37 [31]	I36 [30]	I35 [29]	I34 [28]	I33 [27]	I32 [26]	I31 [25]	I30 [24]	
	+4H	グループ 4								出力データのモニタ用
		I47 [39]	I46 [38]	I45 [37]	I44 [36]	I43 [35]	I42 [34]	I41 [33]	I40 [32]	
	+5H	グループ 5								
		I57 [47]	I56 [46]	I55 [45]	I54 [44]	I53 [43]	I52 [42]	I51 [41]	I50 [40]	
	+6H	グループ 6								
		I67 [55]	I66 [54]	I65 [53]	I64 [52]	I63 [51]	I 62 [50]	I61 [49]	I60 [48]	
	+7H	グループ 7								
		I77 [63]	I76 [62]	I75 [61]	I74 [60]	I73 [59]	I72 [58]	I71 [57]	I70 [56]	
	+1000H	ハンドシェイクステータス入力								
		不定				INT	IBF	STB	ACK	
	+1001H	(使用不可)								
	+1002H	ビットデータ								
		0	0	0	0	0	0	0	BDT	
	+1003H	(使用不可)								
	+1004H	グループデータ								
		PD7	PD6	PD5	PD4	PD3	PD2	PD1	PD0	
	+1005H	(使用不可)								
	+1006H	割り込みコントロール								
		不定		IN1	IN0	IE	IR	IS1	IS0	
	+1007H	設定時間								
		不定			ST4	ST3	ST2	ST1	ST0	

表中のI××は信号名を示します。コネクタの端子番号は図3.1と同じです。
また、[ ]内の数字は、各ビットの番号を示します。
ビット単位の入力を行うときは、このビット番号を使用します。

図4.2 入力ポート



表中のO××は信号名を示します。コネクタの端子番号は図3.2と同じです。
また、[ ]内の数字は、各ビットの番号を示します。
ビット単位の出力を行うときは、このビット番号を使用します。

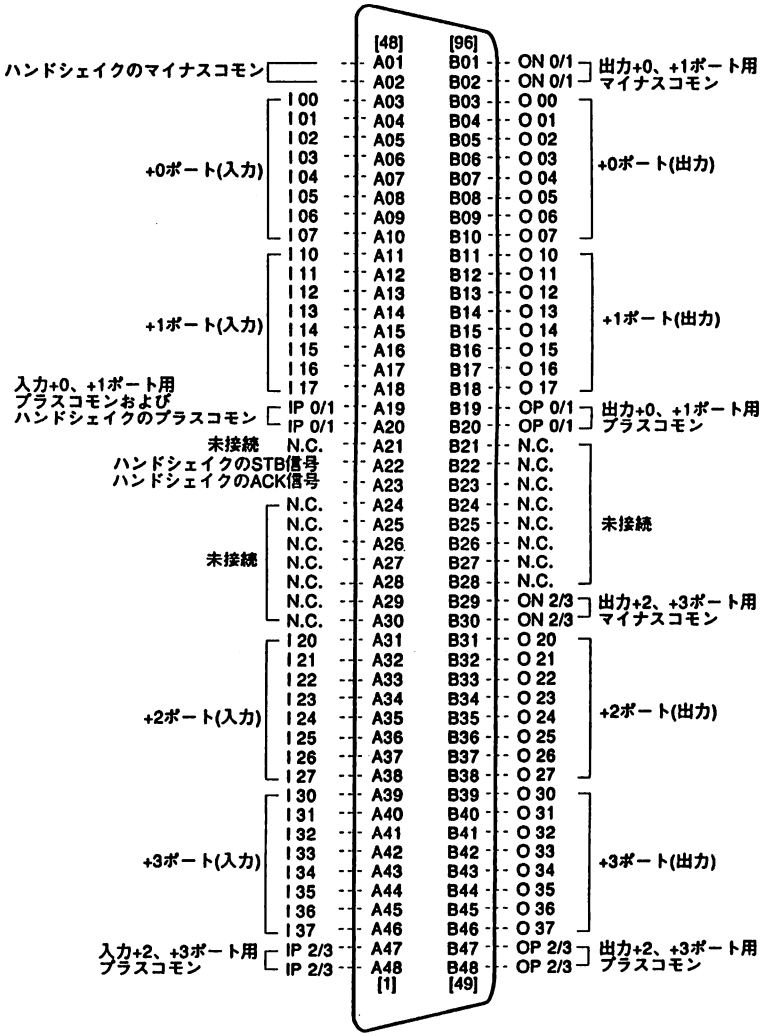
図4.3 出力ポート

▼注意

先頭I/Oアドレス+1001Hにデータを出力しないでください。誤動作の原因になります。

◆インターフェイスコネクタの信号配置<オプション機能>

本インターフェイスボードと外部装置の接続は、ボードに実装された96ピンのコネクタで行います。



・[ ]内は本多通信工業(株)指定の端子番号です。

図4.4 インターフェイスコネクタの信号配置

オプション機能の使用方法

◆ビット単位の入出力機能

この機能を使用すると、任意の1点(1ビット)に対してデータを入出力することができます。

・入力ポート

		D7	D6	D5	D4	D3	D2	D1	D0
先頭I/O アドレス	+1002H	ビットデータ							
		0	0	0	0	0	0	0	PD0

・出力ポート

先頭I/O アドレス	+1002H	ビットデータ							
		0	0	0	0	0	0	0	PD0
	+1003H	ビット選択							
		0	0	BS5	BS4	BS3	BS2	BS1	BS0

図4.5　ビット単位のデータ入出力用ポート

■データを入力する場合

①出力ポートの「ビット選択」ポートに入力したいビットのビット番号を出力してください(ビット番号の指定)。

ビット番号：0～63(3FH)

②次に、入力ポートの「ビットデータ」ポートを入力してください。「1」または「0」のデータが得られます。

プログラム例

ビット番号「31(1FH)」(A46ピン)のデータを入力する場合は、次のようになります。ただし、ボードの先頭I/Oアドレスは、PORT%またはportです。

①BASIC(MS-DOS版)

```
OUT PORT%+&H1003,&H1F
BDT%=INP(PORT%+&H1002)
```

②Microsoft C/C++(MS-DOS版)

```
outp(port+0x1003,0x1F);
bit_data=inp(port+0x1002);
```

■データを出力する場合

①出力ポートの「ビット選択」ポートに、出力したいビットのビット番号を出力してください(ビット番号の指定)。

ビット番号：32(20H)～63(3FH)

②次に、出力ポートの「ビットデータ」ポートに、「1」または「0」を出力してください。

プログラム例
ビット番号「63(3FH)」(B46ピン)に対してデータ「1」を出力する場合は、次のようになります。ただし、ボードの先頭I/Oアドレスは、PORT%またはportです。

①BASIC(MS-DOS版)

```
OUT PORT%+&H1003,&H3F
OUT PORT%+&H1002,1
```

②Microsoft C/C++(MS-DOS版)

```
outp(port+0x1003,0x3F);
outp(port+0x1002,1);
```

◆グループ単位の入出力機能

この機能を使用すると、任意の1グループ(1バイト)に対してデータを入出力することができます。

・入力ポート

先頭I/Oアドレス	D7	D6	D5	D4	D3	D2	D1	D0
+1004H	グループデータ							
	PD7	PD6	PD5	PD4	PD3	PD2	PD1	PD0

・出力ポート

先頭I/Oアドレス								
+1004H	グループデータ							
	PD7	PD6	PD5	PD4	PD3	PD2	PD1	PD0
+1005H	グループ選択							
	0	0	0	0	0	PS2	PS1	PS0

図4.6　グループ単位の入出力用ポート

■データを入力する場合

①出力ポートの「グループ選択」ポートに、入力したいグループのグループ番号を出力してください(グループ番号の指定)。

グループ番号：0～7

②次に、入力ポートの「グループデータ」ポートを入力してください。「0～255(FFH)」のデータが得られます。

プログラム例

グループ番号「0」のデータを入力する場合は、次のようになります。ただし、ボードの先頭I/OアドレスはPORT%またはportです。

①BASIC(MS-DOS版)

```
OUT PORT%+&H1005,0
GDT%=INP(PORT%+&H1004)
```

②Microsoft C/C++(MS-DOS版)

```
out(port+0x1005,0);
group_data=inp(port+0x1004);
```

■データを出力する場合

①出力ポートの「グループ選択」ポートに、出力したいグループのグループ番号を出力してください(グループ番号の指定)。

グループ番号：4～7

②次に、出力ポートの「グループデータ」ポートに、データを出力してください。

出力データ：0～255(FFH)

プログラム例

グループ番号「4」に対してデータ「FFH」を出力する場合は、次のようになります。
ただし、ボードの先頭I/Oアドレスは、PORT%またはportです。

①BASIC(MS-DOS版)

```
OUT PORT%+&H1005,4
OUT PORT%+&H1004,&HFF
```

②Microsoft C/C++(MS-DOS版)

```
outp(port+0x1005,4);
outp(port+0x1004,0xFF);
```

◆デジタルフィルタ機能

この機能を使用すると、すべての入力端子にデジタルフィルタをかけることができます。このため、入力信号のノイズやチャタリングを防止することができます。

デジタルフィルタの動作原理

CPUクロックのサンプリングタイムでレベルチェックを行い、デジタルフィルタの設定時間以上同じ信号レベルが続いたときに入力信号とみなし、パソコンの信号レベルを変化させます。

したがって、設定時間より速い周波数でレベル変化があったときは、そのレベル変化は無効になります。

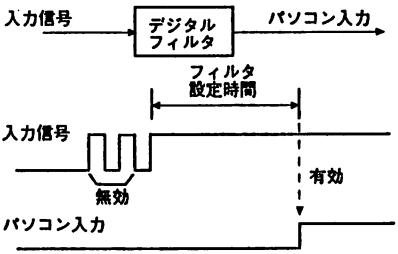


図4.7　デジタルフィルタの動作原理

・入力ポート

先頭I/O アドレス +1007H

D7	D6	D5	D4	D3	D2	D1	D0
設定時間							
不定			ST4	ST3	ST2	ST1	ST0

・出力ポート

先頭I/O アドレス +1007H

時間設定							
0	0	0	ST4	ST3	ST2	ST1	ST0

図4.8　デジタルフィルタ用I/Oポート

■デジタルフィルタの時間を設定する場合

出力ポートの「時間設定」ポートに、設定データ(ST4～ST0)を出力してください。

設定データ：0～20(14H)

「0」を設定したときは、この機能は動作しません。また、電源投入時は「0」が設定されます。なお、デジタルフィルタの時間を設定すると、すべての入力端子にフィルタがかかります。特定の入力端子だけにフィルタをかけることはできません。

▼注意

設定データに上記の範囲外の値を設定しないでください。誤動作の原因となります。

デジタルフィルタの時間と設定データの関係を図4.9に示しますので、設定データを算出し、「時間設定」ポートに出力してください。

$$\text{デジタルフィルタの時間[sec]} = \frac{1}{\text{I/O拡張バスのシステムクロック}} \times 2^{n}$$

n: 設定データ(ST4～ST0)

図4.9 デジタルフィルタの時間と設定データ

DOS/VパソコンのI/O拡張バスのシステムクロックは8MHzです。(ISAバスの場合)詳細は、各パソコンのマニュアルを参照してください。また、表にシステムクロック8MHzの場合の設定データとデジタルフィルタ時間を示します。

表4.1 デジタルフィルタの設定時間

設定データ	デジタルフィルタ時間	設定データ	デジタルフィルタ時間
01H	0.25μs	0BH	256μs
02H	0.5μs	0CH	512μs
03H	1μs	0DH	1ms
04H	2μs	0EH	2ms
05H	4μs	0FH	4ms
06H	8μs	10H	8ms
07H	16μs	11H	16ms
08H	3μs	12H	32ms
09H	64μs	13H	64ms
0AH	128μs	14H	128ms

プログラム例

CPUのクロック周波数が8MHzのパソコンを使用して、デジタルフィルタの時間を8msecに設定する場合、次のようになります。ただし、ボードの先頭I/OアドレスはPORT%または、portです。

①BASIC(MS-DOS版)

```
OUT PORT%+&H1007,&H10
```

②Microsoft C/C++(MS-DOS版)

```
outp(port+0x1007,0x10);
```

■デジタルフィルタに設定した時間を確認する場合

入力ポートの「設定時間」ポートを入力して、設定されているデータ(ST4～ST0)を確認してください。

この確認したデータとデジタルフィルタの時間の関係は、前述の図4.9の通りです。

プログラム例

①BASIC(MS-DOS版)

```
TIM%=INP(PORT%+&H1007)
```

②Microsoft C/C++(MS-DOS版)

```
set_time=inp(port+0x1007);
```

◆出力データのモニタ機能

この機能を使用すると、現在出力されているデータの状態を出力データに影響を与えることなく読み込むことができます。

先頭I/Oアドレス		D7	D6	D5	D4	D3	D2	D1	D0
	+4H	グループ 4							
		I47 [39]	I46 [38]	I45 [37]	I44 [36]	I43 [35]	I42 [34]	I41 [33]	I40 [32]
	+5H	グループ 5							
		I57 [47]	I56 [46]	I55 [45]	I54 [44]	I53 [43]	I52 [42]	I51 [41]	I50 [40]
	+6H	グループ 6							
		I67 [55]	I66 [54]	I65 [53]	I64 [52]	I63 [51]	I62 [50]	I61 [49]	I60 [48]
	+7H	グループ 7							
		I77 [63]	I76 [62]	I75 [61]	I74 [60]	I73 [59]	I72 [58]	I71 [57]	I70 [56]

図4.10　出力データのモニタ用入力ポート

■直接入力ポートをアクセスする場合

データを出力した出力ポートと同じI/Oアドレスを入力してください。

プログラム例

先頭I/Oアドレス+4ポートに出力したデータをモニタする場合、次のようになります。

①BASIC(MS-DOS版)

```
OUT PORT%+4,&HAA
MDT%+INP(PORT%=4)
```

②Microsoft C/C++(MS-DOS版)

```
outp(port+4,0xaa)
m_data=inp(port+4)
```

■任意の1点(1ビット)の出力データを確認する場合

「ビット単位の入出力機能」を参照してください。

■任意の1グループ(8点単位)の出力データを確認する場合

「グループ単位の入出力機能」を参照してください。

◆ハンドシェイク機能

この機能を使用すると、外部からのSTB信号によって割り込みを発生させることができます。

・入力ポート

		D7	D6	D5	D4	D3	D2	D1	D0
先頭I/O アドレス	+1000H	ハンドシェイクステータス入力							
		不定				INT	IBF	STB	ACK

・出力ポート

先頭I/O アドレス	+1000H	ACK信号出力							
		0	0	0	0	0	0	0	ACK

図4.11　ハンドシェイク用I/Oポート

▼注意

先頭I/Oアドレス+1000HにF0Hを出力することにより、ハンドシェイク機能はリセット(初期状態)されます。

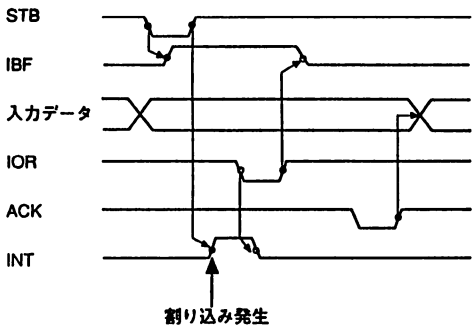


図4.12　ハンドシェイクの入力タイミング

プログラム例(ステータスチェックによるハンドシェイク入力)

①BASIC(MS-DOS版)

```
PORT%=&H300                    'Base I/O address setting
OUT PORT%+&H1000,&HF0          'Handshake function initialization
A=INP(PORT%+&H1000)            'Status input
A=A AND 8
IF(A=0) THEN 30                'INT flag check
DATA%=INP(PORT%)               'Data input
OUT PORT%+&H1000,0             'ACK output
OUT PORT%+&H1000,1
```

②Microsoft C/C++(MS-DOS版)

```
#include<stdio.h>

void main(void)
{
      int a,data,port=0x300;        /*Base I/O address setting*/
      outp(port+0x1000,0xf0);       /*Handshake function initialization*/

      do {
            a=inp(port+0x1000);     /*Status input*/
            a=a&8;
      }while(a==0);                 /*INT flag check*/

      data=inp(port);               /*Data input*/
      outp(port+0x1000,0);          /*ACK output*/
      outp(port+0x1000,1);
}
```

◆割り込みコントロール機能

この機能を使用すると、JP1の割り込み出力端子a~dの内容をコントロールすることができます。

・入力ポート

先頭I/Oアドレス		D7	D6	D5	D4	D3	D2	D1	D0
	+1006H	割り込みコントロール							
		不定		IN1	IN0	IE	IR	IS1	IS0

・出力ポート

先頭I/Oアドレス		割り込みコントロール							
	+1006H	0	0	IN1	IN0	IE	IR	IS1	IS0

図4.13 割り込みコントロール用I/Oポート

■割り込みをコントロールする場合

出力ポートの「割り込みコントロール」ポートにデータを出力することによって、次の4点をコントロールすることができます。

・割り込みの要因
・割り込み入力の信号レベル(論理の反転)
・割り込み入力の一時的禁止
・割り込み出力端子

各ビットの内容を図4.14に示します。

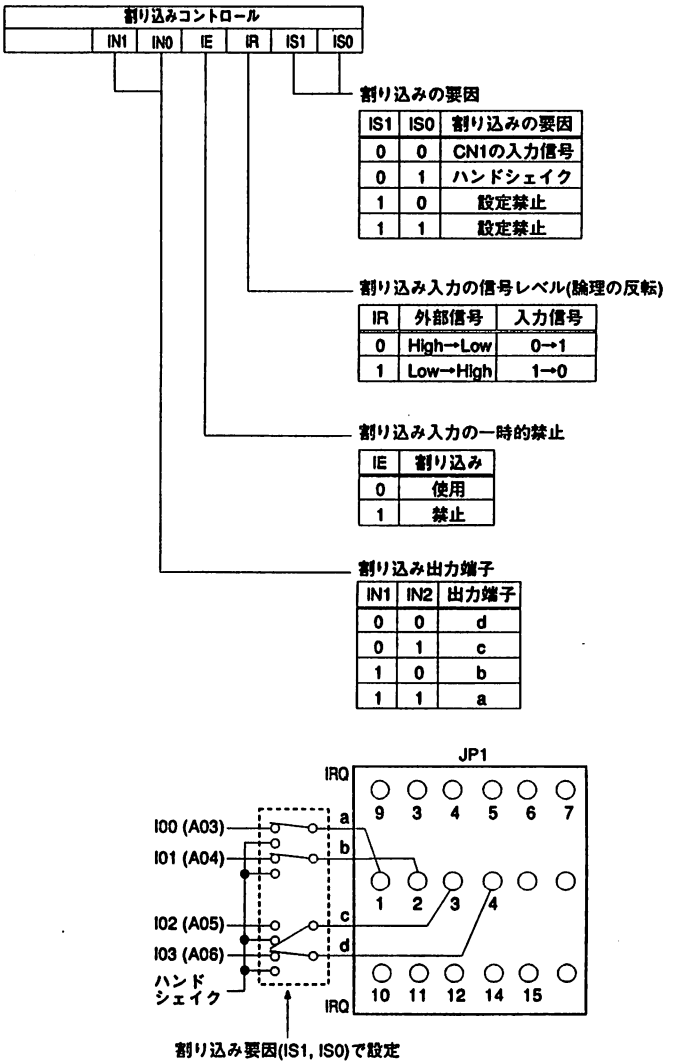


図4.14 「割り込みコントロール」ポートのビット内容

▼注意

①電源投入時は全ビットとも「0」です。

②「割り込み入力の一時的禁止」を設定すると、出力端子a～dのレベルはHighレベルになります。この設定を行う前の状態がLowレベルのときは、割り込みが発生しますので注意してください。

③割り込みを全く使用しないときは、JP1のショートコネクタを抜いておいてください。

プログラム例

次に示す内容で割り込みをコントロールする場合は、下記のようになります。

・割り込みの要因　　：ハンドシェイク
・割り込み入力の信号レベル　：Low→High
・割り込み入力の一時的禁止　：使用
・割り込み出力端子　　：a

ただし、ボードの先頭I/Oアドレスは、PORT%またはportです。

①BASIC(MS-DOS版)

```
OUT PORT%+&H1006,&H35
```

②Microsoft C/C++(MS-DOS版)

```
outp(port+0x1006,0x35);
```

■割り込みコントロールの内容を確認する場合

入力ポートの「割り込みコントロール」ポートを入力して、各ビットの状態を確認してください。各ビットの内容は、前述の図4.14を参照してください。

第5章

ボード本体の仕様

●回路ブロック図…5-2
●仕様…5-3

回路ブロック図

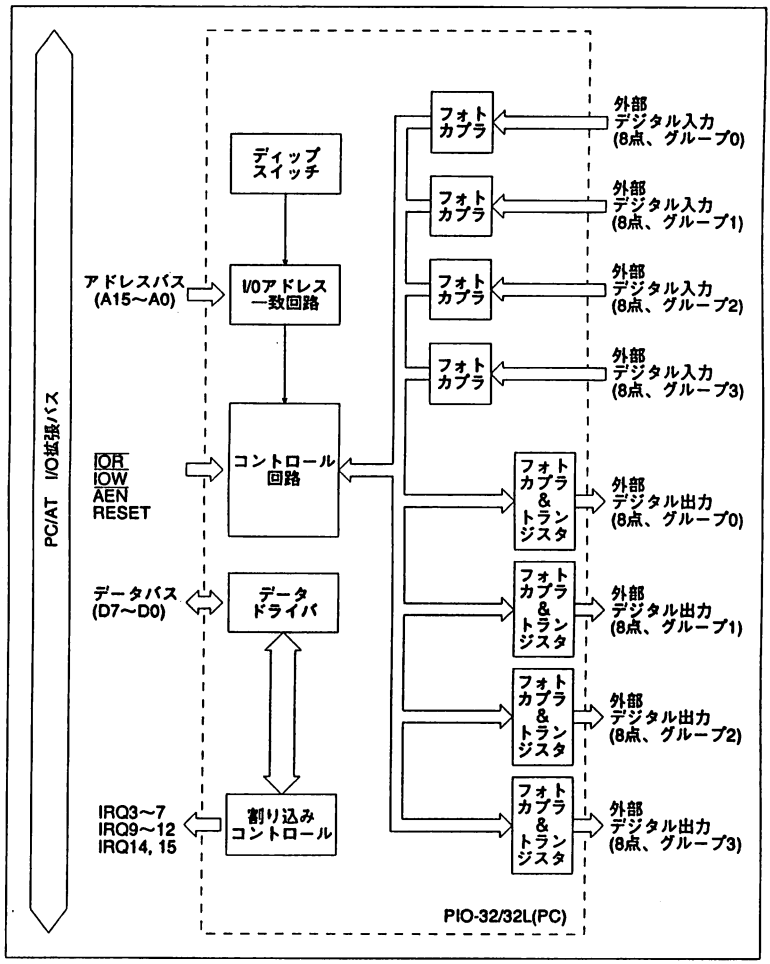


図5.1 回路ブロック図

仕様

ボードの仕様を表5.1に示します。

表5.1　仕様

	項　目		仕　様
入力部	入力形式		フォトカプラ絶縁による電流ソースタイプ入力(負論理)
	入力抵抗		3kΩ
	入力ON電流		3.4mA以上
	入力OFF電流		0.16mA以下
	入力信号の点数		32点(4点は割り込みに使用可能) 注)コモンは16点単位で1コモン
	応答時間		1msec以内
出力部	出力形式		フォトカプラ絶縁による電流シンクタイプ出力(負論理)
	定格	出力耐圧	最大　DC35V
		出力電流	最大　150mA(1点当たり)　注)1コモン当たり最大2A
	出力信号の点数		32点　　注)コモンは16点単位で1コモン
	応答時間		1msec以内
共通部	I/Oアドレス		<汎用入出力機能使用時> 8ビット×4ポート占有 <オプション機能使用時> 8ビット×16ポート占有
	割り込み		IRQ3～7、9～12、14、15のうちいずれか(最大で同時4点まで) <汎用入出力機能使用時> High→Lowのエッジで割り込み発生 <オプション機能使用時> High→LowまたはLow→Highのエッジ (ソフトウェアで設定)で割り込み発生
	外部回路電源		DC12V～24V(±15%) 注)入力1点当たり4mA/12V～8mA/24V
	消費電流		DC5V　180mA(MAX)
	使用条件		0～50℃、20～90%(ただし、結露しないこと)
	信号延長可能距離		50m程度(配線環境による)
	外形寸法		160.0×122.0×18.5mm
	ボード本体の重量		200g

付　録

A. PC/ATシリーズでの割り込み…【2】
B. LSIに対するリカバリタイム…【15】
C. サンプルプログラム…【16】
D. サージ電圧の対策…【33】
E. 実装方法…【34】

A. PC/ATシリーズでの割り込み

当社のボードの中には、パソコンの持つ割り込み機能を利用することができるものがあります。これらのボードで割り込み機能を利用する場合、以下の説明を参考にして適切な処理をしてください。

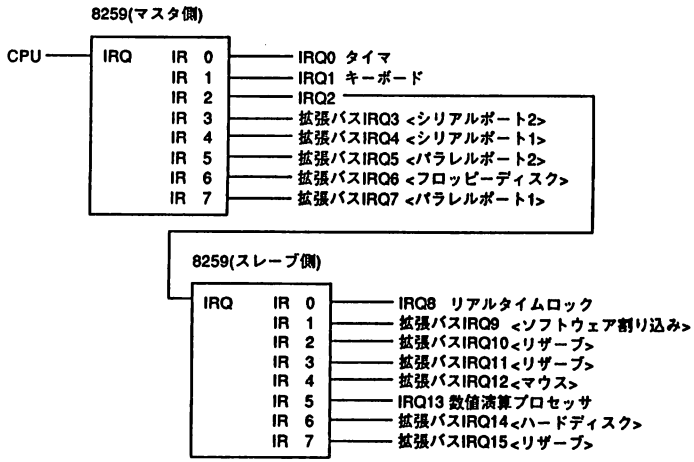
PC/ATとその互換機では、IRQ3～7、9～12、14、15の割り込みレベルの信号が用意されています(XTバスはIRQ2～7となります)。
拡張バスの割り込みレベルは、それぞれの周辺装置に割り当てが決められていますが、そのシステム構成で未使用の割り込みレベルをボードが使用することができます。

なお、PC/ATとその互換機の割り込み機能を使用するためには、パソコンの環境設定処理、割り込み環境復旧処理、および割り込みハンドラ内の応答処理が必要です。

◆割り込みレベルと割り込みベクタ

■割り込みレベル

PC/ATとその互換機には2つの割り込みコントローラ(8259)が使用され、マスタスレーブ構成をとっています。図A.1に示すように、拡張バスのIRQ3～7がマスタ側に割り付けられ、IRQ9～12、14、15がスレーブ側に割り付けられています。割り込み機能を利用するプログラムはこれらの割り込みコントローラを制御するため、使用する割り込みレベル、またマスタ側とスレーブ側では処理の内容が異なります。



＊拡張バスの割り込みレベルに割り当てられる、通常の用途・周辺機器を
< >内に示しています。

図A.1 割り込みコントローラ

▼注意

・PC/ATとその互換機では、IRQ9を使用することができます。IRQ2は使用できません。

・割り込みコントローラは、ボードからの割り込み要求信号のLOWからHIGHへの立ち上がりで割り込みを受け付けます。

・パソコン本体内や他の周辺機器で、すでに使用されている割り込みレベルは使用することができません。

・複数枚のボードで同一の割り込みレベルを使用することはできません。

■割り込みベクタ

割り込みレベルとそれに対応する割り込み処理ルーチンを関係づけるものが割り込みベクタテーブルです。ベクタテーブルはメモリの最下位番地(ゼロ番地)から1KBの大きさがあり、各割り込み処理ルーチンの開始番地は4バイト単位で格納されています。割り込みコントローラに対応する割り込みベクタテーブルのアドレスとベクタ番号の割り当ては、次のとおりです。

表A.1　割り込みベクタテーブルのアドレスとベクタ番号の割り当て

	アドレス	ベクタ番号	用途	
マスタ側	0020H～0023H	08H	タイマ	(IRQ0)
	0024H～0027H	09H	キーボード	(IRQ1)
	0028H～002BH	0AH		(IRQ2)
	002CH～002FH	0BH	シリアルポート2	(IRQ3)
	0030H～0033H	0CH	シリアルポート1	(IRQ4)
	0034H～0037H	0DH	パラレルポート2	(IRQ5)
	0038H～003BH	0EH	フロッピーディスク	(IRQ6)
	003CH～003FH	0FH	パラレルポート1	(IRQ7)
スレーブ側	01C0H～01C3H	70H	リアルタイムクロック	(IRQ8)
	01C4H～01C7H	71H	ソフトウェア割り込み	(IRQ9)
	01C8H～01CBH	72H	リザーブ	(IRQ10)
	01CCH～01CFH	73H	リザーブ	(IRQ11)
	01D0H～01D3H	74H	マウス	(IRQ12)
	01D4H～01D7H	75H	数値演算プロセッサ	(IRQ13)
	01D8H～01DBH	76H	ハードディスク	(IRQ14)
	01DCH～01DFH	77H	リザーブ	(IRQ15)

◆割り込み機能を利用するための処理

ボードからの割り込み要求に対応して割り込みハンドラを実行させるには、パソコンの割り込み環境を割り込みハンドラに合わせるための設定処理が必要です。また、割り込みハンドラが実行されたときには、割り込みコントローラに対して所定の応答処理を行います。割り込みの処理がすべて終了した後は、割り込み環境の復旧処理を行い、パソコンの割り込み環境を元の状態に戻します。

それぞれの処理の詳細を以下に示します。

(1)割り込み環境の設定処理

①割り込み環境の設定処理の詳細

割り込み環境の設定処理には、次の2つの処理が必要です。
・割り込みベクタテーブルの設定
・割り込みコントローラの設定

■割り込みベクタテーブルの設定

現状の割り込みベクタテーブルの内容の保持と割り込みハンドラの割り込みベクタアドレス(開始アドレス)の設定を行います。割り込み環境を復旧するときのために、使用する割り込みレベルに対応した現状の割り込みベクタテーブルの内容を保持しておきます。続いて、割り込みベクタテーブルに割り込みハンドラのベクタアドレスを設定します。

■割り込みコントローラの設定

割り込みコントローラのIMR(Interrupt Mask Register)のマスク状態の保持とIMRのマスク解除、ISR(Interrupt Service Register)のクリアを行います。

割り込み環境を復旧するときのために、現状のIMRのマスク状態を保持しておきます。その後、使用する割り込みレベルに対応したビットのマスクを解除し、そのレベルの割り込みを使用可能にします。続いて、ISRをクリアすることにより、現在割り込みコントローラに受け付けられている割り込み要求をクリアし、あらためて割り込み要求信号の受け付けを開始させます。割り込みレベルと割り込みコントローラのデータの関係を表A.2に示します。

表A.2　割り込みレベルと割り込みコントローラのデータ

	ベクタ番号	割り込み信号	IMRのマスク解除		ISRのクリア	
			ポートアドレス	データ	ポートアドレス	データ
マスタ側	0AH	IRQ2	0021H	FBH	0020H	62H
	0BH	IRQ3	0021H	F7H	0020H	63H
	0CH	IRQ4	0021H	EFH	0020H	64H
	0DH	IRQ5	0021H	DFH	0020H	65H
	0EH	IRQ6	0021H	BFH	0020H	66H
	0FH	IRQ7	0021H	7FH	0020H	67H
スレーブ側	71H	IRQ9	00A1H	FDH	00A0H	61H
	72H	IRQ10	00A1H	FBH	00A0H	62H
	73H	IRQ11	00A1H	F7H	00A0H	63H
	74H	IRQ12	00A1H	EFH	00A0H	64H
	76H	IRQ14	00A1H	BFH	00A0H	66H
	77H	IRQ15	00A1H	7FH	00A0H	67H

割り込みベクタテーブルに割り込みハンドラのベクタアドレスを設定するときと、IMRのマスク解除、ISRのクリアの途中で割り込みが発生するのを防止するため、あらかじめ割り込みを禁止しておきます。設定処理がすべて終了した後、割り込みの許可を行います。

マスク
他のビット・パターン中にある特定のビット位置の情報を変更したり、隔離したりするための文字あるいはビットのパターンのことです。

IMR(割り込みマスクレジスタ)
マスクすべき割り込みラインのビットを格納しています。よって、IMRのセットされているビットに対応したIR入力の割り込み要求は保留されます。

ISR(割り込みサービスレジスタ)
サービス中の割り込みレベルを格納するために用いられます。そして、割り込みの終了コマンドが発せられると更新されます。

②割り込み環境の設定処理の例

■マスタ側(IRQ5)の場合(Microsoft Macro Assembler)

IRQ5のベクタ番号(0dh)を指定し、MS-DOSのファンクションコールでベクタテーブルの内容を取得し、変数に格納します。割り込み禁止後、MS-DOSのファンクションコールでベクタテーブルの内容を割り込みハンドラのベクタアドレスに変更します。

割り込みコントローラのマスタ側のポートアドレス(0021h)から現状のIMRのマスク状態を入力し、変数に格納します。IRQ5に対応するビットのマスクを解除するため、(DFh)との論理積をとってから出力し、IRQ5を使用可能にします。続いて、割り込みコントローラのマスタ側のポートアドレス(0020h)に(65h)を出力し、ISRをクリアします。設定処理がすべて終了した後、割り込みの許可を行います。

割り込みレベルをIRQ2, 3, 4, 6, 7に変更する場合は、表A.2を参照して斜体字の部分を変更してください。

```
mov   al, 0dh                  ; ﾍﾞｸﾀ番号の指定
mov   ah, 35h                  ; 現状の割り込みﾍﾞｸﾀの取得
int   21h                      ; MS-DOS SYSTEM CALL
mov   orgvect_off, bx          ; ｵﾌｾｯﾄｱﾄﾞﾚｽの格納
mov   orgvect_seg, es          ; ｾｸﾞﾒﾝﾄｱﾄﾞﾚｽの格納

cli                            ; 割り込みの禁止

mov   ds, inthandler_segment   ; 割り込みﾊﾝﾄﾞﾗのｾｸﾞﾒﾝﾄｱﾄﾞﾚｽ
mov   dx, inthandler_offset    ; 割り込みﾊﾝﾄﾞﾗのｵﾌｾｯﾄｱﾄﾞﾚｽ
mov   al, 0dh                  ; ﾍﾞｸﾀ番号の指定
mov   ah, 25h                  ; 割り込みﾍﾞｸﾀの変更
int   21h                      ; MS-DOS SYSTEM CALL

in    al, 0021h                ; 現状のﾏｽﾀ側IMRのﾏｽｸ状態の入力
mov   orgimr_ma, al            ; 現状のﾏｽﾀ側IMRのﾏｽｸ状態の格納
and   al, 0dfh                 ; 対応するﾋﾞｯﾄのﾏｽｸ解除
out   0021h, al                ; ﾏｽﾀ側IMRのﾏｽｸ解除

mov   al, 65h                  ; ﾏｽﾀ側ISRのｸﾘｱﾃﾞｰﾀの指定
out   0020h, al                ; ﾏｽﾀ側ISRのｸﾘｱ

sti                            ; 割り込みの許可
```

■マスタ側(IRQ5)の場合(Microsoft C/C++)

現状の割り込みベクタを保持するため、interrupt属性のポインタ変数を宣言します。_dos_getvect( )関数でIRQ5のベクタ番号(0dh)を指定し、ベクタテーブルの内容を取得し、変数に格納します。割り込み禁止後、_dos_setvect( )関数でベクタテーブルの内容を割り込みハンドラのベクタアドレスに変更します。

割り込みコントローラのマスタ側のポートアドレス(0021h)から現状のIMRのマスク状態を入力し、変数に格納します。IRQ5に対応するビットのマスクを解除するため、(DFh)との論理積をとってから出力し、IRQ5を使用可能にします。続いて、割り込みコントローラのマスタ側のポートアドレス(0020h)に(65h)を出力し、ISRをクリアします。設定処理がすべて終了した後、割り込みの許可を行います。

割り込みレベルをIRQ2, 3, 4, 6, 7に変更する場合は、表A.2を参照して斜体字の部分を変更してください。

```
void  _interrupt_far inthandler(void);          /*割り込みﾊﾝﾄﾞﾗのﾌﾟﾛﾄﾀｲﾌﾟ宣言*/
void  (_interrupt_far * orgvect) (void);        /*割り込みﾍﾞｸﾀ格納用ﾎﾟｲﾝﾀ変数*/

orgvect = _dos_getvect(0x0d);                   /*現状の割り込みﾍﾞｸﾀの取得*/
_disable( );                                    /*割り込みの禁止*/
_dos_setvect(0x0d, inthandler);                 /*割り込みﾍﾞｸﾀの変更*/
outp(0x0021, (orgimr_ma=inp(0x0021))&0xdf);     /*現状のﾏｽﾀ側IMRの格納とﾏｽｸ解除*/
outp(0x0020, 0x65);                             /*ﾏｽﾀ側ISRのｸﾘｱ*/
_enable( );                                     /*割り込みの許可*/
```

▼注意

Microsoft Cの旧バージョンでは割り込みの関数が使えないものがあります。ランタイムライブラリリファレンスで確認してください。

■スレーブ側(IRQ12)の場合(Microsoft Macro Assembler)

スレーブ側の割り込みレベル(IRQ9, 10, 11, 12, 14, 15)はマスタ側の割り込みコントローラを経由してCPUに接続されているため、マスタ側とスレーブ側の両方の割り込みコントローラに対して設定します。

IRQ12のベクタ番号(74h)を指定し、MS-DOSのファンクションコールでベクタテーブルの内容を取得し、変数に格納します。割り込み禁止後、MS-DOSのファンクションコールでベクタテーブルの内容を割り込みハンドラのベクタアドレスに変更します。

割り込みコントローラのマスタ側を使用可能にした後、スレーブ側のポートアドレス(00A1h)から現状のIMRのマスク状態を入力し、変数に格納します。IRQ12に対応するビットのマスクを解除するため、(EFh)との論理積をとってから出力し、IRQ12を使用可能にします。続いて、割り込みコントローラのマスタ側ISRをクリアした後、スレーブ側のポートアドレス(00A0h)に(64h)を出力し、ISRをクリアします。設定処理がすべて終了した後、割り込みの許可を行います。

割り込みレベルをIRQ9, 10, 11, 14, 15に変更する場合は、表A.2を参照して斜体字の部分を変更してください。

```
mov   al, 74h                 ; ﾍﾞｸﾀ番号の指定
mov   ah, 35h                 ; 現状の割り込みﾍﾞｸﾀの取得
int   21h                     ; MS-DOS SYSTEM CALL
mov   orgvect_off, bx         ; ｵﾌｾｯﾄｱﾄﾞﾚｽの格納
mov   orgvect_seg, es         ; ｾｸﾞﾒﾝﾄｱﾄﾞﾚｽの格納

cli                           ; 割り込みの禁止

mov   ds, inthandler_segment  ; 割り込みﾊﾝﾄﾞﾗのｾｸﾞﾒﾝﾄｱﾄﾞﾚｽ
mov   dx, inthandler_offset   ; 割り込みﾊﾝﾄﾞﾗのｵﾌｾｯﾄｱﾄﾞﾚｽ
mov   al, 74h                 ; ﾍﾞｸﾀ番号の指定
mov   ah, 25h                 ; 割り込みﾍﾞｸﾀの変更
int   21h                     ; MS-DOS SYSTEM CALL

in    al, 21h                 ; 現状のﾏｽﾀ側IMRのﾏｽｸ状態の入力
mov   orgimr_ma, al           ; 現状のﾏｽﾀ側IMRのﾏｽｸ状態の格納
and   al, 0fbh                ; 対応するﾋﾞｯﾄのﾏｽｸ解除
out   0021h, al               ; ﾏｽﾀ側IMRのﾏｽｸ解除

in    al, 00a1h               ; 現状のｽﾚｰﾌﾞ側IMRのﾏｽｸ状態の入力
mov   orgimr_su, al           ; 現状のｽﾚｰﾌﾞ側IMRのﾏｽｸ状態の格納
and   al, 0efh                ; 対応するﾋﾞｯﾄのﾏｽｸ解除
out   00a1h, al               ; ｽﾚｰﾌﾞ側IMRのﾏｽｸ解除

mov   al, 62h                 ; ﾏｽﾀ側ISRのｸﾘｱﾃﾞｰﾀの指定
out   0020h, al               ; ﾏｽﾀ側ISRのｸﾘｱ

mov   al, 64h                 ; ｽﾚｰﾌﾞ側ISRのｸﾘｱﾃﾞｰﾀの指定
out   00a0h, al               ; ｽﾚｰﾌﾞ側ISRのｸﾘｱ

sti                           ; 割り込みの許可
```

■スレーブ側(IRQ12)の場合(Microsoft C/C++)

スレーブ側の割り込みレベル(IRQ9, 10, 11, 12, 14, 15)はマスタ側の割り込みコントローラを経由してCPUに接続されているため、マスタ側とスレーブ側の両方の割り込みコントローラに対して設定します。

現状の割り込みベクタを保持するため、interrupt属性のポインタ変数を宣言します。_dos_getvect( )関数でIRQ12のベクタ番号(74h)を指定し、ベクタテーブルの内容を取得し、変数に格納します。割り込み禁止後、_dos_setvect( )関数でベクタテーブルの内容を割り込みハンドラのベクタアドレスに変更します。

割り込みコントローラのマスタ側を使用可能にした後、スレーブ側のポートアドレス(00A1h)から現状のIMRのマスク状態を入力し、変数に格納します。IRQ12に対応するビットのマスクを解除するため、(EFh)との論理積をとってから出力し、IRQ12を使用可能にします。続いて、割り込みコントローラのマスタ側ISRをクリアした後、スレーブ側のポートアドレス(00A0h)に(64h)を出力し、ISRをクリアします。設定処理がすべて終了した後、割り込みの許可を行います。

割り込みレベルをIRQ9, 10, 11, 14, 15に変更する場合は、表A.2を参照して斜体字の部分を変更してください。

```
void  _interrupt_far inthandler(void);      /*割り込みﾊﾝﾄﾞﾗのﾌﾟﾛﾄﾀｲﾌﾟ宣言*/
void  (_interrupt_far * orgvect)(void);     /*割り込みﾍﾞｸﾀ格納用ﾎﾟｲﾝﾀ変数*/

orgvect = _dos_getvect(0x74);               /*現状の割り込みﾍﾞｸﾀの取得*/
_disable( );                                /*割り込みの禁止*/
_dos_setvect(0x74, inthandler);             /*割り込みﾍﾞｸﾀの変更*/
outp(0x0021,(orgimr_ma=inp(0x0021))&0xfb);  /*現状のﾏｽﾀ側IMRの格納とﾏｽｸ解除*/
outp(0x00a1,(orgimr_su=inp(0x00a1))&0xef);  /*現状のｽﾚｰﾌﾞ側IMRの格納とﾏｽｸ解除*/
outp(0x0020, 0x62);                         /*ﾏｽﾀ側ISRのｸﾘｱ*/
outp(0x00a0, 0x64);                         /*ｽﾚｰﾌﾞ側ISRのｸﾘｱ*/
_enable( );                                 /*割り込みの許可*/
```

(2)割り込み環境の復旧処理

①割り込み環境の復旧処理の詳細

割り込み環境の復旧処理には、次の2つの処理が必要です。
・割り込みコントローラの復旧
・割り込みベクタテーブルの復旧

■割り込みコントローラの復旧

割り込みコントローラのIMRのマスク状態を解除する前の状態に戻します。解除する前に保持しておいた、IMRのマスク状態を再設定します。割り込みベクタテーブルの再設定とIMRのマスクの復旧の途中で割り込みが発生するのを防止するため、あらかじめ割り込みを禁止しておきます。設定処理がすべて終了した後、割り込みの許可を行います。

■割り込みベクタテーブルの復旧

割り込みベクタテーブルの内容を、変更する前の状態に戻します。設定する前に保持しておいた、使用する割り込みレベルに対応した割り込みベクタテーブルの内容を再設定します。

②割り込み環境の復旧処理の例

■マスタ側(IRQ5)の場合(Microsoft Macro Assembler)

割り込み禁止後、割り込みコントローラのマスタ側のポートアドレス(0021h)に変更前のIMRのマスク状態を出力します。IRQ5の割り込みベクタ番号(0dh)を指定し、MS-DOSのファンクションコールでベクタテーブルの内容を変更前の値に戻します。復旧処理がすべて終了した後、割り込みの許可を行います。

割り込みレベルをIRQ2, 3, 4, 6, 7に変更する場合は、表A.2を参照して斜体字の部分を変更してください。

```
cli                             ; 割り込みの禁止

mov   al, orgimr_ma             ; 変更前のﾏｽﾀ側IMRのﾏｽｸ状態
out   0021h, al                 ; ﾏｽﾀ側IMRのﾏｽｸの復旧

mov   ds, orgvect_seg           ; 変更前のｾｸﾞﾒﾝﾄｱﾄﾞﾚｽ
mov   dx, orgvect_off           ; 変更前のｵﾌｾｯﾄｱﾄﾞﾚｽ
mov   al, 0dh                   ; ﾍﾞｸﾀ番号の指定
mov   ah, 25h                   ; 割り込みﾍﾞｸﾀの復旧
int   21h                       ; MS-DOS SYSTEM CALL

sti                             ; 割り込みの許可
```

■マスタ側(IRQ5)の場合(Microsoft C/C++)

割り込み禁止後、割り込みコントローラのマスタ側のポートアドレス(0021h)に変更前のIMRのマスク状態を出力します。_dos_setvect( )関数でベクタテーブルの内容を変更前の値に戻します。復旧処理がすべて終了した後、割り込みの許可を行います。

割り込みレベルをIRQ2, 3, 4, 6, 7に変更する場合は、表A.2を参照して斜体字の部分を変更してください。

```
_disable( );                        /*割り込みの禁止*/
outp(0x0021, orgimr_ma);            /*ﾏｽﾀ側IMRのﾏｽｸの復旧*/
_dos_setvect(0x0d, orgvect);        /*割り込みﾍﾞｸﾀの復旧*/
_enable( );                         /*割り込みの許可*/
```

■スレーブ側(IRQ12)の場合(Microsoft Macro Assembler)

割り込み禁止後、割り込みコントローラのスレーブ側のポートアドレス(00A1h)に変更前のIMRのマスク状態を出力します。IRQ12の割り込みベクタ番号(74h)を指定し、MS-DOSのファンクションコールでベクタテーブルの内容を変更前の値に戻します。復旧処理がすべて終了した後、割り込みの許可を行います。

割り込みレベルをIRQ9, 10, 11, 14, 15に変更する場合は、表A.2を参照して斜体時の部分を変更してください。

```
cli                                 ; 割り込みの禁止

mov   al, orgimr_su                 ; 変更前のｽﾚｰﾌﾞ側IMRのﾏｽｸ状態
out   00A1h, al                     ; ｽﾚｰﾌﾞ側IMRのﾏｽｸの復旧

mov   ds, orgvect_seg               ; 変更前のｾｸﾞﾒﾝﾄｱﾄﾞﾚｽ
mov   dx, orgvect_off               ; 変更前のｵﾌｾｯﾄｱﾄﾞﾚｽ
mov   al, 74h                       ; ﾍﾞｸﾀ番号の指定
mov   ah, 25h                       ; 割り込みﾍﾞｸﾀの復旧
int   21h                           ; MS-DOS SYSTEM CALL

sti                                 ; 割り込みの許可
```

■スレーブ側(IRQ12)の場合(Microsoft C/C++)

割り込み禁止後、割り込みコントローラのスレーブ側のポートアドレス(00A1h)に変更前のIMRのマスク状態を出力します。_dos_setvect( )関数でベクタテーブルの内容を変更前の値に戻します。復旧処理がすべて終了した後、割り込みの許可を行います。

割り込みレベルをIRQ9, 10, 11, 14, 15に変更する場合は、表A.2を参照して斜体字の部分を変更してください。

```
_disable( );                        /*割り込みの禁止*/
outp(0x00a1, orgimr_su);            /*ｽﾚｰﾌﾞ側IMRのﾏｽｸの復旧*/
_dos_setvect(0x74, orgvect);        /*割り込みﾍﾞｸﾀの復旧*/
_enable( );                         /*割り込みの許可*/
```

(3)割り込みハンドラの処理

①割り込みハンドラの処理の詳細

割り込みハンドラは、次の処理で構成されます。
・上位レベルの割り込みの許可
・レジスタの退避
・割り込みハンドラの本来の処理
・割り込みコントローラへの応答処理
・レジスタの復旧
・割り込みハンドラからの復帰

割り込みハンドラを実行する最初に上位レベルの割り込みの許可を行います。この処理は、他のプログラムの動作を考慮して、割り込みハンドラ実行中でも上位レベルの割り込みが受け付けられるようにしておくためです。割り込みハンドラ内で使用するレジスタの退避を行ってから、割り込みハンドラ本来の処理を行います。割り込みハンドラ本来の処理が終了したら、割り込みコントローラに対して応答処理を行い、割り込み処理が終了したことを通知します。最後に使用したレジスタを復旧してから、割り込みハンドラを実行する前の処理に戻ります。

②割り込みハンドラの処理の例

■Microsoft Macro Assemblerの場合

```
sti             ;  上位レベルの割り込みの許可
push  ax        ;  ﾚｼﾞｽﾀの退避
push  bx
push  cx
push  dx
push  si
push  es
割り込みハンドラの本来の処理
割り込みコントローラへの応答処理

pop   es        ;  ﾚｼﾞｽﾀの復旧
pop   si
pop   dx
pop   cx
pop   bx
pop   ax
iret            ;  割り込みﾊﾝﾄﾞﾗからの復帰
```

■Microsoft C/C++の場合

```
void  _interrupt_far inthandler(void)
{
      _enable( );          /*上位ﾚﾍﾞﾙの割り込みの許可*/
      割り込みハンドラの本来の処理
      割り込みコントローラへの応答処理


}
```

(4)割り込みコントローラへの応答処理

割り込みハンドラ本来の処理が終了したら、割り込みコントローラに対して応答処理を行い、割り込み処理が終了したことを通知します。この処理は、使用する割り込みレベルがマスタ側かスレーブ側かで処理の内容が異なります。

①マスタ側の割り込みレベルを使用した場合

マスタ側の割り込みレベル(IRQ2～7)を使用した場合は、マスタ側の割り込みコントローラに対してのみ処理を行います。割り込みコントローラにEOI(End Of Interrupt)を送り、現在処理している割り込み処理に対応するビットをリセットします。

■応答処理の例(Microsoft Macro Assembler)

```
mov   al, 20h        ; EOIのﾃﾞｰﾀ指定
out   0020h, al      ; ﾏｽﾀ側へのEOIの送出
```

■応答処理の例(Microsoft C/C++)

```
outp(0x0020, 0x20);  /*ﾏｽﾀ側へのEOIの送出*/
```

EOI (End Of Interrupt)
割り込み処理の終了を通知し、割り込みサービス状態をリセットします。

②スレーブ側の割り込みレベルを使用した場合

スレーブ側の割り込みレベル(IRQ9, 10, 11, 12, 14, 15)はマスタ側の割り込みコントローラを経由してCPUに接続されているため、マスタ側とスレーブ側の両方の割り込みコントローラに対して応答処理を行います。それぞれの割り込みコントローラにEOI (End Of Interrupt)を送り、現在処理している割り込み処理に対応する割り込みコントローラのビットをリセットします。

ただし、現在処理している割り込み以外のスレーブ側の割り込み要求が残っている場合はマスタ側の割り込みコントローラに対しての応答処理は行いません。他の割り込み要求が残っているかどうかを判断し、所定の処理を行います。

■応答処理の例(Microsoft Macro Assembler)

```
    mov   al, 20h        ; EOIのﾃﾞｰﾀ指定
    out   00a0h, al      ; ｽﾚｰﾌﾞ側へのEOIの送出
    in    al, 02efh      ; 割り込みｺﾝﾄﾛｰﾗのﾘｶﾊﾞﾘﾀｲﾑの確保

    mov   al, 0bh        ; ｽﾚｰﾌﾞ側ISRのﾃﾞｰﾀ入力ｺﾏﾝﾄﾞ指定
    out   00a0h, al      ; ｽﾚｰﾌﾞ側へのｺﾏﾝﾄﾞ送出

    in    al, 02efh      ; 割り込みｺﾝﾄﾛｰﾗのﾘｶﾊﾞﾘﾀｲﾑの確保

    in    al, 00a0h      ; ｽﾚｰﾌﾞ側ISRのﾃﾞｰﾀ入力
    or    al, al         ; 他の割り込み要求の確保
    jnz   PEND

    mov   al, 20h        ; EOIのﾃﾞｰﾀ指定
    out   0020h, al      ; ﾏｽﾀ側へのEOIの送出

PEND:
```

■応答処理の例(Microsoft C/C++)

```
outp(0x00a0, 0x20);      /*ｽﾚｰﾌﾞ側へのEOIの送出*/
rt=inp(0x02ef);          /*割り込みｺﾝﾄﾛｰﾗのﾘｶﾊﾞﾘﾀｲﾑの確保*/
outp(0x00a0, 0x0b);      /*ｽﾚｰﾌﾞ側ISRのﾃﾞｰﾀ入力ｺﾏﾝﾄﾞ送出*/
if (inp(0x00a0) == 0)    /*ｽﾚｰﾌﾞ側ISRのﾃﾞｰﾀ入力と他の割り込み要求の確認*/
    outp(0x0020, 0x20); /*ﾏｽﾀ側へのEOIの送出*/
```

＊空ポートをinすることにより、最小0.5μsecのリカバリタイムが確保されます。
上記例ではCOM4のスクラッチレジスタをリードしています。

B. LSIに対するリカバリタイム

パソコンのCPUは動作クロックの高速化に伴い、周辺LSIをソフトウェアで制御するのに制限がでてきました。当社のボードに搭載されているLSIの中で、特に表B.1に示されているデバイスに対してアクセスする場合、以下の説明を参考にして適切な処理を行ってください。

i386CPUのパソコンまではソフトウェアによるウエイト(JMP $+2)で、LSIにアクセスしたときのリカバリタイムを確保できました。しかし、i486のCPU以降のパソコンではCPUのキャッシュメモリ機能によって、ソフトウェアによるウエイトではリカバリタイムを確保できません。

そこで、i486以降のCPUで、このリカバリタイムを確保する方法のひとつを次に説明します。

PC/ATとその互換機では、I/Oアドレス2EFhポート(COM4のスクラッチレジスタ)を1回INすることによって、最小0.5μsecの時間がかかります。この方法ではCPUのタイプと動作速度に依存しないため、この時間をリカバリタイムとして利用できます。表B.1に示されているデバイスに対してアクセスしたあとに、2EFhポートに必要な回数のIN命令を行い、リカバリタイムを確保してください。

表B.1 LSIにアクセスしたあとに必要な2EFhポートのINの回数

LSI名称	出力時	入力時
i8237相当品	0回	0回
i8254相当品	1回	1回
i8255相当品	1回	1回
i8259相当品	1回	1回
NS16550	1回	1回
μPD7210C	1回	0回

■プログラム記述例(i8254相当品へのアクセス例)

・Microsoft Macro Assembler

```
OUT   DX, AL        ; i8254相当品へのアクセス
IN    AL, 2EFH      ; リカバリタイムとして IN  AL, 2EFH を1回実行
```

・Microsoft C/C++

```
outp(port, byte); /*i8254相当品へのアクセス*/
rt=inp(0x2ef);    /*リカバリタイムとして rt=inp(0x2ef); を1回実行*/
```

C. サンプルプログラム

◆Q-BASIC版入力サンプルプログラム

このプログラムは、Q-BASICで記述されています。実行しますと、各ポートからデータを入力し、その値を16進数の形式で画面上に表示します。

■仕様

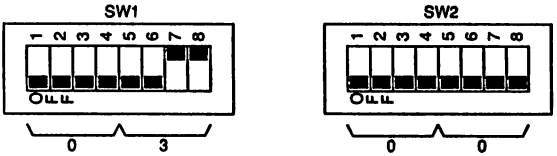
0300H, 0301H, 0302H,および0303Hポートからデータを入力し、ディスプレイに表示します。

■準備

本プログラムを動作させるためには、次の準備が必要です。

①割り込みは、使用しませんのでJP1のショートコネクタはすべて抜いてください。

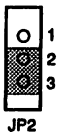
②ポートアドレスの設定は、下図のようにディップスイッチ(SW1, SW2)で0300Hに設定してください。



■印がスイッチのツマミを表します。

図C.1 ディップスイッチの設定

③JP2は下図のとおり、工場出荷時設定(2-3間をショート)の状態で使用してください。



▨印がジャンパを表します。

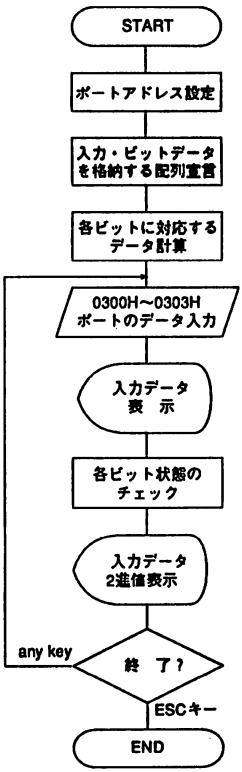
図C.2 機能切り換えジャンパの設定

■表示画面とフローチャート

①表示画面

```
**********************************
*      Byte Data Input Sample     *
**********************************

PORT      INPUT DATA    D7 D6 D5 D4 D3 D2 D1 D0
0300H         AAH        1  0  1  0  1  0  1  0
0301H         55H        0  1  0  1  0  1  0  1
0302H         80H        1  0  0  0  0  0  0  0
0303H         19H        0  0  0  1  1  0  0  1

CONTINUE    ---> PRESS ANY KEY
END         ---> PRESS ESC KEY
```

②フローチャート



図C.3 入力サンプルプログラムのフローチャート

```
'      ************************************
'      *     Sample Program               *
'      *          data input    Ver 1.0   *
'      ************************************

    CLS :    WIDTH 80, 25
    DIM IN.DATA(4), BIT(8)
    SETPORT = &H300
    LOCATE 3, 20: PRINT "************************************"
    LOCATE 4, 20: PRINT "*     Byte Data Input Sample     *"
    LOCATE 5, 20: PRINT "************************************"

    FOR J = 0 TO 7: BIT(J) = 2 ^ J: NEXT J
LOOP1:
    LOCATE 9, 17: PRINT "port     input data   D7 D6 D5 D4 D3 D2 D1
D0"
    LOCATE 19, 25: PRINT "End       ---> Push ESC key"
    LOCATE 18, 25: PRINT "Input     ---> Push any key"
WAIT1:
    X$ = INKEY$: IF X$ = "" THEN GOTO WAIT1 ELSE IF ASC(X$) = &H1B
THEN GOTO PEND

'     *********************
'     *    Input  data    *
'     *********************

    FOR I = 0 TO 3
        IN.DATA(I) = INP(SETPORT + I)
        LOCATE I + 10, 17
        PRINT USING "&          &       "; RIGHT$("000" + HEX$(SETPORT +
I), 4); RIGHT$("0" + HEX$(IN.DATA(I)), 2);
        FOR J = 7 TO 0 STEP -1
            IF (IN.DATA(I) AND BIT(J)) <> 0 THEN PRINT "  1";  ELSE
PRINT "  0";
        NEXT J
    NEXT I
    LOCATE 18, 25: PRINT "Continue ---> Push any key"
WAIT2:
    X$ = INKEY$: IF X$ = "" THEN GOTO WAIT2 ELSE IF ASC(X$) = &H1B
THEN GOTO PEND

    FOR I = 10 TO 13: LOCATE I, 17: PRINT SPACE$(60): NEXT I
    GOTO LOOP1
PEND:
    END
```

◆Microsoft C版入力サンプルプログラム

このプログラムは,「◆Q-BASIC版入力サンプルプログラム」と同じ仕様のものがMicrosoft Cで記述されています。スイッチ類は「◆Q-BASIC版入力サンプルプログラム」と同じ設定にしてください。表示画面とフローチャートも同様です。

```
/*******************************/
/*        SAMPLE PROGRAM       */
/*   MS-C DATA INPUT  VER 1.00 */
/*******************************/

#include<stdio.h>
#include<conio.h>
#include<dos.h>
#include<math.h>
#define cls()           printf("\033[2J")
#define locate(x,y)     printf("\033[%d;%dH", y+1, x+1)

main()
{
    unsigned char a;
    unsigned portadd = 0x0300;
    unsigned char A[4], Bit[8];
    char HOW_DO = '1';
    char i,j;

    cls();

    locate(20,3);  printf("**********************************\n");
    locate(20,4);  printf("**     Byte Data Input Sample    *\n");
    locate(20,5);  printf("**********************************\n");

    locate(17,9);
    printf("port    input data   D7 D6 D5 D4 D3 D2 D1 D0\n");

    while (HOW_DO != 0x1b)
    {
        locate(25,18);
        printf("Input     ---->   Push any key\n");
        locate(25,19);
        printf("End       ---->   Push ESC key\n");
        for (i=0; i<=4; i++)
            {
            locate(16,10+i);
            printf("                                                  ");
            }
        HOW_DO = getch();

        if (HOW_DO != 0x1b)
        {
        for (i=0; i<=3; i++)
            {
            a = 0x80;
            A[i] = inp (portadd + i);
            locate(17,10+i);
            printf("%xH          %xH\n", portadd +i, A[i]);
            for (j=7; j>=0; j--)
                {
                if ((A[i] & a) > 0) Bit[j] = 1;
```

```
                    else Bit[j] = 0;
                locate(39+(7-j)*3,10+i);
                printf("%x  ",Bit[j]);
                a = (a>>1);
                }
            }
            locate(25,18);
            printf("Continue ---->   Push any key\n");
            HOW_DO = getch();
        }
    }
    cls();
}
```

◆Q-BASIC版出力サンプルプログラム

このプログラムは、Q-BASICで記述されています。実行しますと、キーボードから16進数のデータを入力後、各ポートからデータを出力し、その値を16進数の形式で画面上に表示します。

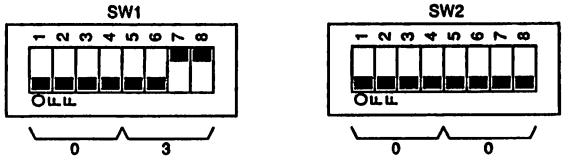
■仕様

0300H, 0301H, 0302H,および0303Hポートへ出力するデータをバイト単位でキーボードから入力した後、それぞれのポートへデータを出力します。

■準備

本プログラムを動作させるためには、次の準備が必要です。

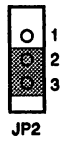
①ポートアドレスの設定は、下図のようにディップスイッチ(SW1、SW2)で0300Hに設定してください。



■印がスイッチのツマミを表します。

図C.4　ディップスイッチの設定

②JP2は下図のとおり、工場出荷時設定(2-3間をショート)の状態で使用してください。



■印がジャンパを表します。

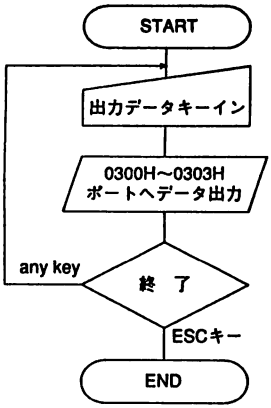
図C.5　機能切り換えジャンパの設定

■表示画面とフローチャート

①表示画面

```
**********************************
*    Byte Data Output Sample     *
**********************************

ENTER OUTPUT DATA FOR 300H PORT (00-FF) --> AA
ENTER OUTPUT DATA FOR 301H PORT (00-FF) --> 55
ENTER OUTPUT DATA FOR 302H PORT (00-FF) --> 80
ENTER OUTPUT DATA FOR 303H PORT (00-FF) --> 19

DATA HAS BEEN OUTPUTTED.

CONTINUE    ---> PRESS ANY KEY
END         ---> PRESS ESC KEY
```

②フローチャート



図C.6 出力サンプルプログラムのフローチャート

```
'      ************************************
'      *          Sample Program          *
'      *      Byte data output   Ver 1.0  *
'      ************************************

    CLS :    WIDTH 80, 25
    LOCATE 3, 20: PRINT "************************************"
    LOCATE 4, 20: PRINT "**      Byte Data Output Sample    **"
    LOCATE 5, 20: PRINT "************************************"

'      ********************************
'      *    Data input  by keyboard   *
'      ********************************

    LOCATE  9, 15: PRINT "ENTER a 2-digit Hex number at 300H port -->
'
    LOCATE 10, 15: PRINT "ENTER a 2-digit Hex number at 301H port -->
'
    LOCATE 11, 15: PRINT "ENTER a 2-digit Hex number at 302H port -->
'
    LOCATE 12, 15: PRINT "ENTER a 2-digit Hex number at 303H port -->
'

WHILE (1)

    PD0$ = ""
    WHILE (LEN(PD0$) < 2)
       LOCATE 9, 60: LINE INPUT PD0$                    '300H port
DATA INPUT
    WEND
    LOCATE 9, 60: PRINT RIGHT$("0" + PD0$, 2); SPACE$(5)
    PD0 = VAL("&H" + RIGHT$(PD0$, 2))

    PD1$ = ""
    WHILE (LEN(PD1$) < 2)
       LOCATE 10, 60: LINE INPUT PD1$                   '301H port
DATA INPUT
    WEND
    LOCATE 10, 60: PRINT RIGHT$("0" + PD1$, 2); SPACE$(5)
    PD1 = VAL("&H" + RIGHT$(PD1$, 2))

    PD2$ = ""
    WHILE (LEN(PD2$) < 2)
       LOCATE 11, 60: LINE INPUT PD2$                   '302H port
DATA INPUT
    WEND
    LOCATE 11, 60: PRINT RIGHT$("0" + PD2$, 2); SPACE$(5)
    PD2 = VAL("&H" + RIGHT$(PD2$, 2))

    PD3$ = ""
    WHILE (LEN(PD3$) < 2)
       LOCATE 12, 60: LINE INPUT PD3$                   '303H port
DATA INPUT
    WEND
    LOCATE 12, 60: PRINT RIGHT$("0" + PD3$, 2); SPACE$(5)
    PD3 = VAL("&H" + RIGHT$(PD3$, 2))


    LOCATE 18, 25: PRINT "Output   ---> Push any key"
```

```
    LOCATE 19, 25: PRINT "End       ---> Push ESC key"
WAIT1:
    X$ = INKEY$: IF X$ = "" THEN GOTO WAIT1 ELSE IF ASC(X$) = &H1B
THEN GOTO PEND
                                    'WAITING PUSH KEY

 '       **********************
 '       *    Output data     *
 '       **********************

    OUT &H300, PD0:                 '300H port's data output
    OUT &H301, PD1:                 '301H port's data output
    OUT &H302, PD2:                 '302H port's data output
    OUT &H303, PD3:                 '303H port's data output

    LOCATE 18, 25: PRINT "Continue ---> Push any key"
WAIT2:
    X$ = INKEY$: IF X$ = "" THEN GOTO WAIT2 ELSE IF ASC(X$) = &H1B
THEN GOTO PEND
                                    'WAITING PUSH KEY
    FOR I = 18 TO 19: LOCATE I, 25: PRINT SPACE$(30): NEXT I      'DE-
LETE COMENT
    FOR I = 9 TO 12: LOCATE I, 60: PRINT SPACE$(10): NEXT I       'DE-
LETE COMENT
WEND
PEND:
   END
```

◆Microsoft C版出力サンプルプログラム

このプログラムは、「◆Q-BASIC版出力サンプルプログラム」と同様のものがMicrosoft Cで記述されています。スイッチ類は、「◆Q-BASIC版出力サンプルプログラム」と同じ設定にしてください。表示画面とフローチャートも同様です。

```
/*********************************/
/*        SAMPLE PROGRAM         */
/*   MS-C DATA OUTPUT  VER 1.00 */
/*********************************/

#include<stdio.h>
#include<conio.h>
#include<dos.h>

#define  off   0
#define  on    1
#define cls()            printf("\033[2J")
#define locate(x,y)      printf("\033[%d;%dH", y+1, x+1)

main()
{
    unsigned portadd = 0x0300;
    unsigned char      pd0,pd1,pd2,pd3;
    int                i;
    char               HOW_DO = '1';

    cls();

    locate(17,3);  printf("**********************************\n");
    locate(17,4);  printf("**     Byte Data Output Sample   *\n");
    locate(17,5);  printf("**********************************\n");

    locate(15, 9);  printf("ENTER a 2-dight Hex number at 300H port --
> ");
    locate(15,10);  printf("ENTER a 2-dight Hex number at 301H port --
> ");
    locate(15,11);  printf("ENTER a 2-dight Hex number at 302H port --
> ");
    locate(15,12);  printf("ENTER a 2-dight Hex number at 303H port --
> ");


    while (HOW_DO != 0x1b)
    {
        locate(25,16); printf("                                    ");
        locate(25,17); printf("                                    ");

        locate(60, 9); scanf("%x",&pd0);
        locate(60,10); scanf("%x",&pd1);
        locate(60,11); scanf("%x",&pd2);
        locate(60,12); scanf("%x",&pd3);

        locate(25,16); printf("Output   ---->   Push any key\n");
        locate(25,17); printf("End      ---->   Push ESC key\n");
        locate( 1,25); HOW_DO = getch();
```

```
            if (HOW_DO != 0x1b)
            {
                outp(portadd  ,pd0);
                outp(portadd+1,pd1);
                outp(portadd+2,pd2);
                outp(portadd+3,pd3);

                locate(25,16); printf("Continue ---->   Push any key\n");
                locate( 1,25); HOW_DO = getch();

                for (i = 9 ; i <= 12 ; i++)
                {
                    locate(60,i);printf("      ");
                }
            }
        }
        cls();
    }
```

◆Q-BASIC版割り込みサンプルプログラム

このプログラムはQ-BASICから機械語プログラム(Microsoft Macro Assemblor)を割り込み処理プログラムとして登録し、使用する例として紹介するものです。

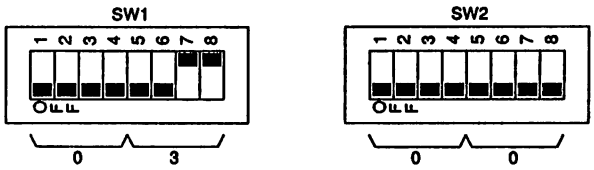
■仕様

割り込み信号(IRQ5)が発生するたびに機械語プログラム側で割り込み回数をカウントし、BASIC側でその値を表示します。

■準備

本サンプルプログラムを動作させるには、次の準備が必要です。

①ポートアドレスの設定は、下図のようにディップスイッチ(SW1、SW2)で0300Hに設定してください。

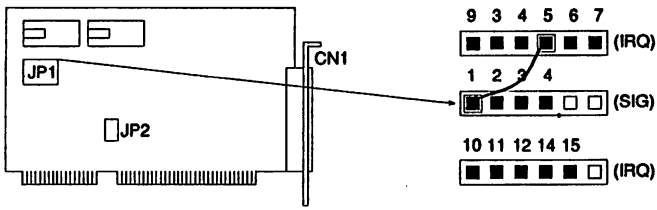


図C.7 ディップスイッチの設定

②JP2は下図のとおり、工場出荷時設定(2-3間をショート)の状態で使用してください。

図C.8 機能切り換えジャンパの設定

③割り込みレベルはIRQ5を使用していますので、JP1はSIG1～IRQ5間をショートしてください。



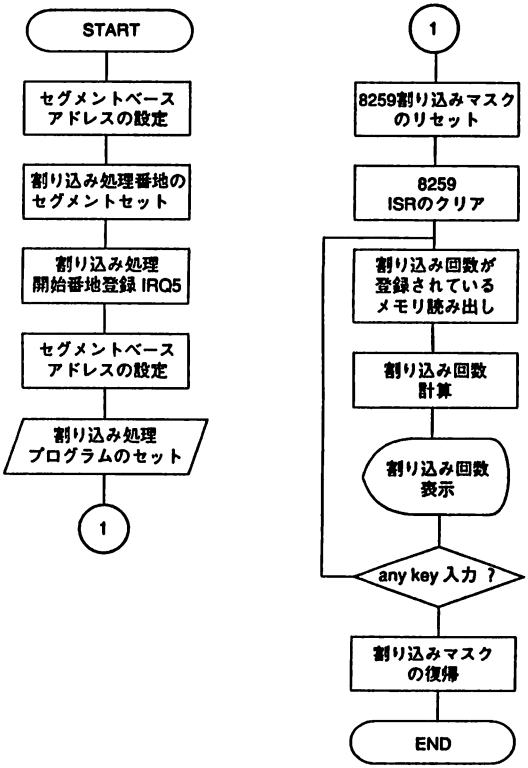
図C.9 割り込みジャンパの設定

■表示画面とフローチャート

①表示画面

```
**************************************
**      Interrupt Sample            **
**************************************

INTERRUPT COUNTER OF IRQ5 = 54

END    ---> PRESS ESC KEY
```

②フローチャート



図C.10　割り込みサンプルプログラムのフローチャート

```
'   **********************************
'   **   INTERRUPT SAMPLE PROGRAM   **
'   **              Ver. 1.00       **
'   **********************************

    DIM MBUFF%(&H190): CLS
    WIDTH 80, 25: KEY OFF

' *****  GET SEGMENT  *****
    LOCATE 3, 20: PRINT "**********************************"
    LOCATE 4, 20: PRINT "**     Interrupt Sample          **"
    LOCATE 5, 20: PRINT "**********************************"
    LOCATE 11, 20: PRINT "Interrupt counter of IRQ5 ="
    LOCATE 18, 20: PRINT "END ---> Push ESC key   "

    MCHSEG = VARSEG(MBUFF%(0))
    SEGH = (INT(MCHSEG AND &HFF00) / &H100) AND &HFF
    SEGL = MCHSEG AND &HFF

' *****  SET INTERRUPT TABLE ADDRESS  *****
    DEF SEG = &H0
    POKE &H34, &H0
    POKE &H35, &H0
    POKE &H36, SEGL
    POKE &H37, SEGH

' *****  STORE MACHINE LANGUAGE  *****
    DEF SEG = MCHSEG
    RESTORE ASM
    FOR I = 0 TO 20
    READ DAT: POKE I, DAT
    NEXT I

ASM:
    DATA &HFB                 : ' STI
    DATA &H1E                 : ' POP    DS
    DATA &H50                 : ' POP    AX
    DATA &H53                 : ' PUSH   BX
    DATA &H8C, &HC8           : ' MOV    AX, CS
    DATA &H8E, &HD8           : ' MOV    DS, AX
    DATA &HBB, &H02, &H03     : ' MOV    BX, 302H
    DATA &HFF, &H07           : ' INC    WORD PTR [BX]
    DATA &HB0, &H20           : ' MOV    AL, 20H
    DATA &HE6, &H20           : ' MOV    20H, AL
    DATA &H5B                 : ' POP    BX
    DATA &H58                 : ' POP    AX
    DATA &H1F                 : ' POP    DS
    DATA &HCF                 : ' IRET

' *****  IRT COUNT SET  *****
    POKE &H303, 0
    POKE &H302, 0

' *****  CLEAR MASK REGISTER  *****
    A = INP(&H21)
    A = (A AND &HDF)
    OUT &H21, A
    OUT &H20, &H65
```

```
 ' *****  LOOP  *****
WHILE (A$ <> CHR$(&H1B))
    A$ = INKEY$
    DEF SEG = MCHSEG
    A = PEEK(&H303) * 256 + PEEK(&H302)
    LOCATE 11, 47: PRINT A
WEND

 ' *****  DISABLE INTERRUPT  *****
    A = INP(&H21)
    A = A OR &H20
    OUT &H21, A

END
```

◆Microsoft C言語版割り込みサンプルプログラム

このプログラムは「◆Q-BASIC版割り込みサンプルプログラム」と同じ仕様のものがMicrosoft Cで記述されています。スイッチ類は、「◆Q-BASIC版割り込みサンプルプログラム」と同じ設定にしてください。表示画面とフローチャートも同様です。

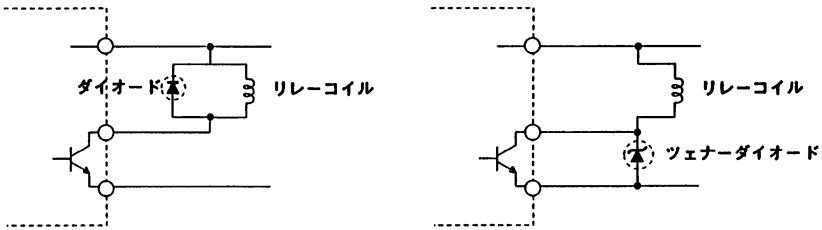
```
/************************************/
/*        SAMPLE PROGRAM            */
/*   MS-C INTERRUPT COUNT  VER 1.00 */
/************************************/

#include<stdio.h>
#include<conio.h>
#include<dos.h>

    static  unsigned  int port = 0x300;
    static  int cnt = 0;
    static  int dat = 0;

#define cls()           printf("\033[2J")
#define locate(x,y)     printf("\033[%d;%dH", y+1, x+1)

void cursor(int m)
{
    union REGS r;
    switch(m)
    {
        case 0:
            r.h.ah = 1;
            r.h.ch = 0x20;
            r.h.cl = 0x20;
            break;
        case 1:
            r.h.ah = 1;
            r.h.ch = 06;
            r.h.cl = 07;
            break;
    }
    int86(0x10,&r,&r);
}

void interrupt far int5_c(void)
{
    _disable();
    cnt++;
    outp(0x20,0x20);
    _enable();
}

void main(void)
{
    int masdat;

    cls();
    cursor(0);
    locate(20,3);  printf("**********************************\n");
    locate(20,4);  printf("**     Interrupt Sample          *\n");
    locate(20,5);  printf("**********************************\n");
```

```
    locate(20,18);
    printf("End ----> Push ESC key");
    locate(20,11);
    printf("Interrupt counter of IRQ5 = 0 ");

    _dos_setvect(0xD,(void(interrupt far *)())int5_c);
    masdat = inp(0x21);
    outp(0x21, masdat & 0xDF);
    outp(0x20, 0x65);

    do
    {
        locate(48,11);
        printf("%d",cnt);
    }while(kbhit() == 0 || getch() != 0x1b);

    outp(0x21,masdat);
    cursor(1);
}
```

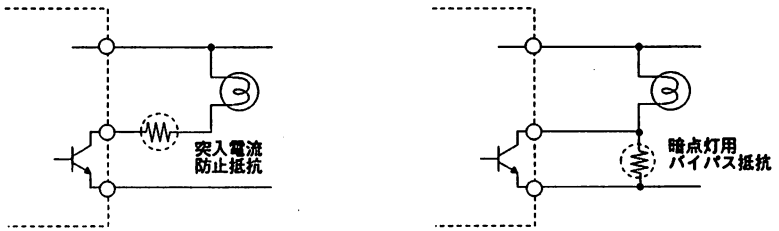
D. サージ電圧の対策

デジタル出力に誘導負荷(リレーコイル)や白熱電球のように、サージ電圧や突入電流が発生する負荷を接続する場合は、出力段の破損防止やノイズによる誤動作防止のため、相応の保護対策が必要です。リレーなどコイルを急速に遮断すると、急激な高電圧パルスが発生します。この電圧が出力トランジスタの耐電圧を超えるとトランジスタの劣化、さらには破損に至ることがあります。そのため、リレーのコイルなど誘導負荷を駆動する場合には、必ずサージ吸収素子を接続してください。以下にサージ電圧対策の例を示します。

●リレーコイル使用例



●ランプ使用例



図D.1 サージ電圧の対策例

▼注意

保護回路を取り付ける場合、負荷および接点のおよそ50cm以内でないと効果が発揮できません。

E. 実装方法

拡張スロットに実装するボードは色々な種類のものがありますが、どの拡張スロットも実装方法は同じです。

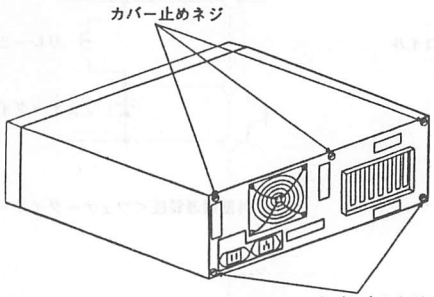
■作業を行う前に

①電源スイッチをOFFにします。

②電源ケーブルをACコンセントから抜きます。

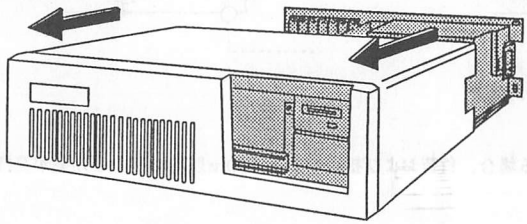
■ボードの実装

①背面カバーを留めているネジを外します。ネジを外す場合は、ネジ頭に合ったドライバを使用してください。

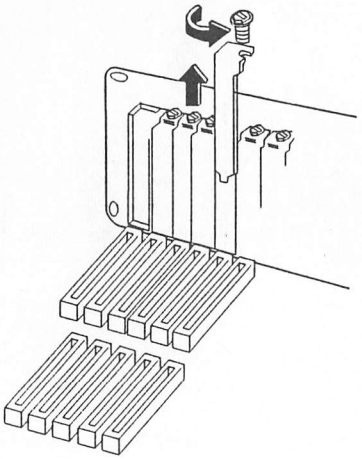


IBM-PC/XT/ATの裏面

②カバー両側を両手で引き出し、本体からカバーを外します。

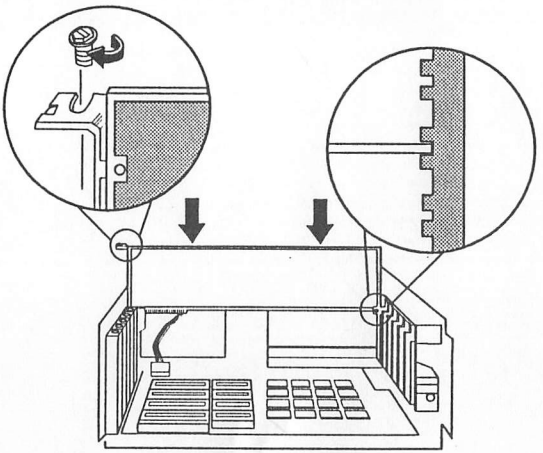


③実装しようとするスロットのブラケット止めネジを外し、ブラケットを背面パネルから取り去ります。



④部品面を右側にしてボードを差し込みます。

⑤ボードを挿入後、背面パネルをネジで固定します。



⑥カバーを元通りにネジ止めし、コネクタ、電源などをセットします。

PIO-32/32L(PC)
解説書

1995年03月31日　制定
1999年02月25日　改訂

発行所　株式会社コンテック

東部支社　〒105-0014　東京都港区芝2-3-3 芝東京海上ビル　TEL　03-3769-1054
西部支社　〒555-0025　大阪市西淀川区姫里3-9-31　TEL　06-6472-0265

PC-MODULE／PC-HELPER
テクニカルインフォメーション

TEL　：03-5443-5304
FAX　：03-5443-5303
E-mail　：p1ougi@tokyo.contec.co.jp

分類番号	A-41-187
コード番号	LZP8381